DATABASE A PROVA DI CRASH

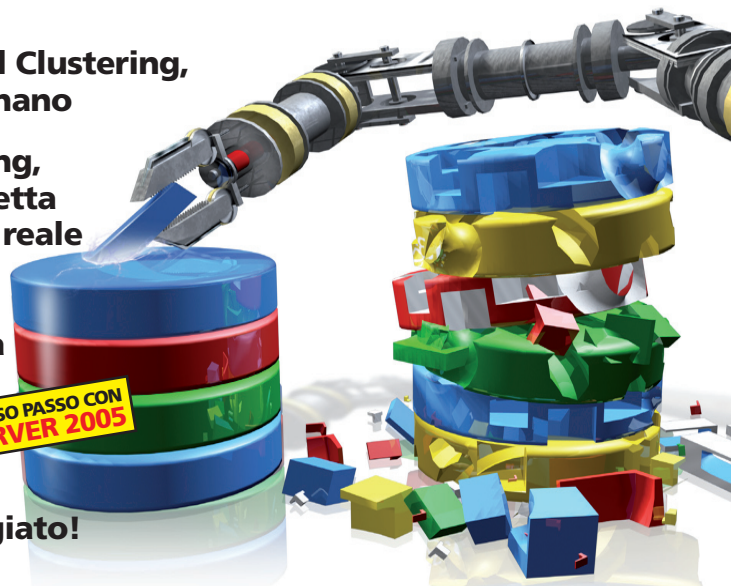
Progetta applicazioni che non smettono di funzionare anche in caso di danni irreparabili come la rottura di un Hard Disk...

TEORIA: Dal Failover al Clustering, cosa sono e come funzionano

TECNICA: Il DB Mirroring, per avere una copia perfetta sempre OnLine in tempo reale

PRATICA: Sviluppa in .NET software che sfrutta connessioni affidabili

RECOVERING: Recupera un database o un file master danneggiato!

GUIDA PASSO PASSO CON SQL SERVER 2005

CREA UN TUO PLUGIN PER GOOGLE DESKTOP

Ecco come estendere l'applicazione più usata del momento aggiungendo un sistema di gestione degli appuntamenti...

NOVITÀ

FATTI IL BOT PER SKYPE

Aggiungi nuove funzioni alla tua chat preferita con ruby e i web services

MOBILE

LE MAPPE NEL PALMARE

Colloca su una cartina un indirizzo presente nella rubrica

INTERNET APPLICATION

STAMPE DAL WEB PERFETTE

Utilizza i CSS per ottenere layout professionali e compatibili con tutte le stampanti

.SQL SERVER

LAVORARE CON SERVICE BROKER

DB non raggiungibile? Salva tutto in una coda e aggiorna quando possibile

DA SQL A XML!

Mai più conversioni difficoltose, ecco come far fare tutto al server

VISUAL BASIC

INVIA E RICEVI FILE SU INTERNET

Sviluppa applicazioni che dialogano fra loro e sfruttano la rete per scambiare i dati

.NET

RIDUCI LE RIGHE DI CODICE

Programmi lunghissimi addio. Arriva il NameSpace MY che ti facilita la vita

JAVA

MENU DINAMICI

Applicazioni accattivanti anche con Java. Come aggiungere icone e grafica

ALLA CONQUISTA DEL DESKTOP

Le nuove funzioni di Java 6. Un esempio completo di software ridotto nella traybar

SQL NO PROBLEM

Impara a usare Hibernate, il tool che trasforma tabelle e dati in oggetti!

ALGORITMI DI GREEDY Ecco come risolvere i problemi procedendo per passi logici ben definiti

Anno X - N.ro 8 (105) - Agosto 2006 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, C. Bellucci, M. Bigatti, R. Brunetti, L. Buono, L. Caputo, D. De Michelis, C. Durante, F. Grimaldi, M. Scala, G. Sudano

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Responsabile grafico di progetto Salvatore Vuono
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti
Direttore Generale: Massimo Rizzo

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 30/09/06
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Assistenza tecnica: ioprogrammo@edmaster.it
Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di luglio 2006

Da Linux a Windows

È successo un fatto strano! O almeno strano in relazione alla tendenza a cui eravamo abituati ad assistere negli ultimi anni. Evolution, il client email più utilizzato in ambito Linux funziona ora anche in ambiente Windows. Se fosse un fatto isolato ci sarebbe poco su cui discutere. Tuttavia il numero di applicazioni che nasce in ambiente Unix e viene portata in ambiente Windows comincia ad essere davvero elevato. Si pensi a Firefox, ad OpenOffice, a Sylpheed, ai vari Apache e company che da tempo producono una doppia versione. D'altra parte è vero anche il contrario, le applicazioni native in ambiente Windows continunano ad essere portate anche su Linux, e sono nati alcuni progetti che hanno addirittura lo scopo di cercare una compatibilità binaria. In questa ottica come dobbiamo adeguarci noi programmatori ad un mondo in cui le barriere, persino fra sistemi operativi, tendono ad assottigliarsi? La parola d'ordine è "flessibilità". Dobbiamo essere in grado, oggi più che mai di progettare codice riusabile, facilmente adattabile alle varie esigenze ed ai vari sistemi operativi. Rimane il dubbio: "Quanto è utile specializzarsi in un'area della programmazione, piuttosto che avere un'infarinatura globale di tutto?" La risposta, come sempre non è assoluta. Molto probabilmente un mercato così flessibile merita una conoscenza di massima di ogni area tecnica, ma è anche vero che ogni programmatore ha il dovere di specializzarsi in un particolare settore quando se ne presenta l'esigenza. La formazione in tutto ciò ricopre, un ruolo essenziale. Ecco perché noi di ioProgrammo investiamo continuamente in nuove tecniche che possano rendere la divugazione dell'informazione ancora più veloce, più facile da apprendere, pur conservando quel carattere di approfondimento che da oltre dieci anni ci caratterizza. Ecco perché accanto alla tradizionale riv ista cartacea, che contiene materiale che può essere "digerito" con la calma, di chi vuole approfondire, affianchiamo anche i video e gli altri materialiche invece rappresentamo un'informazione più veloce, più pronta da essere usata subito, ancora una volta al fianco di chi prima di essere un programmatore, è un appassionato di tecnologia.

Fabio Farnesi ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

DATABASE A PROVA DI CRASH

Progetta applicazioni che non smettono di funzionare anche in caso di danni irreparabili come il Crash di un HD

- ✔ **TECNICA: il database mirroring per avere una copia perfetta sempre OnLine in tempo reale**
- ✔ **TEORIA: Dal FailOver al Clustering, cosa sono e come funzionano?**
- ✔ **PRATICA: costruire applicazioni che non si interrompono mai**
- ✔ **RECOVERING: Come recuperare un DB o un file master danneggiato**

pag. 12

CREA UN TUO PLUGIN PER GOOGLE DESKTOP

Ecco come estendere l'applicazione più usata del momento aggiungendo un sistema di gestione degli appuntamenti

pag. 62

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

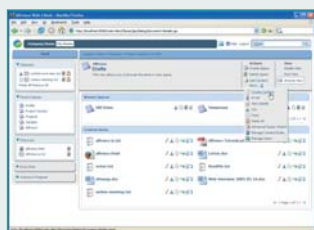
http://forum.ioprogrammo.it

Prodotti del mese

Alfresco 1.2.1

Il CMS secondo JSP

Semplicità è la parola d'ordine per accedere alla logica operativa di Alfresco, un innovativo sistema di content management che sta riscontrando un certo successo presso gli sviluppatori di tutto il mondo. Padre di Alfresco è John Newton, un nome che tra i "consumatori" di CMS dovrebbe suscitare una certa reazione, poiché stiamo parlando del cofondatore di Documentum, tra i primissimi sistemi di content management che già negli anni '90 spopolava presso la nicchia degli addetti ai lavori. A distanza di tre lustri, le cose sono cambiate anche nel mondo CMS, dove, il concetto di portale come spazio virtuale in cui ospitare contenuti dalla natura dinamica si è consolidato, e Alfresco ha saputo mantenere il passo con i tempi rimanendo uno dei più completi CMS del Web.

[pag.107]

DotNetNuke 4.3.0

Un framework per la costruzione di CMS

Era nato come un semplice CMS ma nel tempo si è evoluto in modo rapidissimo ed è diventato un CMS strutturato in modo piramidale. Chi vuole può installarlo come semplice sistema di Document Management, ed in questo senso potrà sfruttare le funzioni esposte e non sono certamente poche. Il layout è molto personalizzabile, i moduli sono molti e facilmente integrati. Coloro i quali invece necessitano di una maggiore personalizzazione possono pensare di mettere le mani direttamente nei sorgenti. Addirittura esiste uno starter kit per Microsoft Visual Studio 2005. Si tratta di un prodotto decisamente maturo ed affidabile

[pag.106]

Drupal 4.7.2

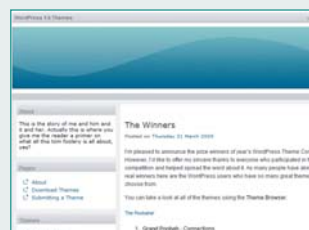
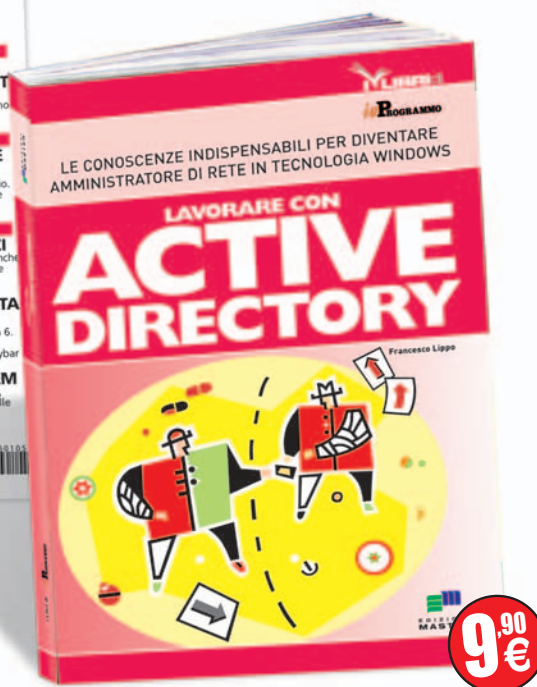
La nuova versione della piattaforma per bloggers

Un blog è un contenitore singolo di informazioni. Una piattaforma per bloggers è un sistema che include in se più di un blog. Esempi concreti possono essere Splender, o Bloggers. Drupal è una piattaforma per bloggers scritta in PHP estremamente efficiente. Le sue caratteristiche più interessanti sono quella della modularità e della flessibilità del layout. Dal punto di vista del programmatore è utile sapere che Drupal espone un intero framework per la costruzione dei moduli. Se avrete occasione di dare uno sguardo ai sorgenti, e agli esempi vi accorgerete che il framework in questione è concepito in maniera assolutamente solida ed è stato progettato in maniera estremamente efficace. Espandere Drupal non richiede particolari sforzi nello sviluppo

[pag.107]

Wordpress

Il primo blog!

All'inizio c'era PHPNuke, poi sono arrivati Xoom e gli altri, e il primo a seguire il modello "Blog" che ormai conosciamo così bene è stato probabilmente Wordpress. Si tratta di un sistema molto semplice e veloce da utilizzare, ma allo stesso tempo anche molto completo ed efficace. Se non volete incorrere in inutili complicazioni, è lo strumento che fa per voi. Wordpress è leggero ma dispone di tutte le funzionalità tipiche di un CMS di grande levatura. Inoltre senza troppi sforzi è possibile far diventare Wordpress un multiblog in cui gli spazi di molti utenti possono condensarsi in un unico centro. In definitiva si tratta di un prodotto affidabile, espandibile e che gode di una community di supporto importante alla quale potrete sempre rivolgervi [pag.107]

RIVISTA + LIBRO + CD-ROM in edicola

I contenuti del libro

Lavorare con Active Directory

Chi non conosce il concetto di Directory? Una directory è un catalogo strutturato di informazioni. Nel caso più comune in una directory sono contenute le informazioni relative ai file che popolano il nostro disco rigido. Ma cosa succederebbe se esistessero Directory che contengono informazioni maggiormente strutturate? ad esempio l'elenco delle stampanti della nostra rete, o le generalità anagrafiche dei dipendenti dell'azienda, o la descrizione di un prodotto o qualunque altro tipo di informazione strutturata? Tutti i client della rete potrebbero accedere a queste informazioni snellendo molto il lavoro dell'amministratore. E' quello che si prefigge di fare il servizio di "Active Directory". In questo libro imparerete cosa deve fare un buon amministratore di rete per sviluppare un network efficiente e funzionale.

LE CONOSCENZE INDISPENSABILI PER DIVENTARE AMMINISTRATORE DI RETE IN TECNOLOGIA WINDOWS

- Introduzione ad Active Directory
- Domini di Windows e concetti base
- Ldap e i client degli utenti
- Esempi pratici

GLI ALLEGATI DI IOPROGRAMMO

▼ Office Application

Word, Excel, Outlook, InfoPath e gli altri programmi della famiglia Office costituiscono, probabilmente, il set di applicazioni più usate di tutti i tempi e in qualunque ambito. Allora perchè non sviluppare personalizzazioni che consentono di integrare le vostre applicazioni direttamente all'interno di Office? Ad esempio, piuttosto che utilizzare un applicativo separato da Excel per recuperare la lista dei fornitori, è possibile fare in modo che Excel si connetta a SQL Server, recuperi i dati, li elabori e li mostri in output sullo stesso Excel. Così come si può redigere un contratto con Word creando un'applicazione che lo elabori automaticamente a partire da una raccolta di leggi preesistenti, e così via. Visual Studio 2005 Tools for Office System (VSTO) è un ambiente che estende Visual Studio 2005 e permette di sviluppare applicazioni integrate all'interno di Office. Grazie a VSTO, sviluppare un'applicazione che giri nel contesto di Word piuttosto che di Excel diventa semplice così come sviluppare un'applicazione .NET tradizionale. La tecnica è innovativa e rivoluzionaria. Seguendo gli MSDN Webcast di questo mese, noterete quanto possa essere utile e semplice usare Visual Studio 2005 Tools for Office System .

I VIDEOCORSI PER PROGRAMMARE BENE

- **Introduzione a Visual Studio 2005 Tools for Office System**
- **Funzionalità avanzate e deployment di applicazioni sviluppate con VSTO 2005**
- **Action Pane, controlli e Smart Tag**
- **Supporto per Outlook e InfoPath**

msdn WEBCAST

INFORMAZIONI SU MSDN WEBCAST
http://www.microsoft.it/msdn/webcast_msdn
http://forum.ioprogrammo.it

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi, che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN.

Come è composto tipicamente un Webcast?

Normalmente viene presentata una serie di slide commentate da un esperto di tecnologie Microsoft. A supporto di queste presentazioni spesso vengono realizzate delle demo in presa diretta che mostrano dal vivo come usare le tecnologie oggetto del Webcast

Come mai nei webcast allegati alla rivista si parla di chat e di altri strumenti interattivi?

La natura dei Webcast è quella di essere seguiti online e in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. E' possibile porre domande in presa diretta al relatore oppure partecipare ai sondaggi che vengono proposti. In questo modo gli sviluppatori possono aggiornarsi e approfondire i temi di loro interesse con maggiore efficacia. I Webcast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i Webcast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei Webcast MSDN direttamente da un CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità dei programmatori italiani con la più ampia gamma di strumenti di formazione e aggiornamento.

Su ioProgrammo troverò tutti Webcast MSDN?

Ne troverai sicuramente una buona parte. Direttamente sul sito MSDN potrai consultare il calendario dei prossimi webcast a cui iscriverti per seguirli in diretta oppure consultare l'archivio delle registrazioni di quelli già realizzati. L'indirizzo per saperne di più è www.microsoft.it/msdn/webcast_msdn, segnalo nel tuo bookmark, non può mancare!

L'iniziativa sarà ripetuta sui prossimi numeri?

Sicuramente sì, alla prossima.

News

IL CODICE DELLA SFIDA

Bello ed intelligente il torneo di programmazione che si è tenuto a Melbourne in un locale chiamato Loop Bar. A sfidarsi sono state ben 14 Crew di programmatori.
L'oggetto della sfida era: " chi scriverà in 10 minuti il miglior codice per il recupero di un'immagine danneggiata". E mentre le Crew si sfidavano sul filo di patch, ottimizzazioni e ricompilazioni, il codice scorreva su un megaschermo sotto la vista di tutti.A far da sottofondo alla competizione, la musica dei Simulus e dei VS Chorus Crew. In molti hanno definito l'evento come emozionante. Ha vinto il team di Toasted Monkeys che ha risolto il quesito in 5 minuti e 40 secondi. Dimostrando che la programmazione può anch essere uno Show!

RILASCIATO OPEN WATCOM 1.5

E' stato per lungo tempo uno dei compilatori maggiormente performanti disponibili sul mercato. Ad oggi rimane l'unico a poter creare in C o in Fortran codice eseguibile per Windows 3.1 ed MS-DOS. Le novità introdotte riguardano l'introduzione dello standard ISO/IEC TR 24731 con la sostituzione delle non sicure strcpy(), strcat(), strncpy() e strncat(), con le più nuove strcpy_s(), strcat_s(), strncpy_s() e strncat_s(), create da Microsoft. Altre modifiche minori sono state introdotte nel compilatore, tuttavia a sorprendere non è tanto la capacità di Open Watcom di continuare a innalzare il suo tasso tecnologico, quanto la sua longevità. Se è vero che I sistemi operativi supportati da questo compilatore sono scomparsi ormai da tempo, è difficile spiegarsi chi siano i reali utilizzatori di questo compilatore. Eppure, per caratteristiche tecniche, per la sua disponibilità in formato Open Source, per le sue spiccate propensioni didattiche, per il ricoprire una nicchia evidentemente attiva, Open Watcom riesce ancora a tener bene il mercato, e persino ad aggiornarsi con nuove versioni.

ARRIVA WINDOWS COMPUTE CLUSTER SERVER 2003

A Microsoft mancava un software in grado di eseguire operazioni parallele High-Performance. Oggi la lacuna è stata colmata, è recente infatti l'annuncio del rilascio della versione RTM (release to manufacturing) di Windows Compute Cluster Server 2003. Il prodotto sarà disponibile già dal mese di Agosto 2006. Gli ambiti applicativi sono I più disparati: dall'ingegneria, alla ricerca medica, all'esplorazione, la dove fino ad ora Unix deteneva un mercato assolutamente senza concorrenza. Non a caso fra I primi clienti annoveriamo il CASPUR, consorzio italiano interuniversitario per le applicazioni di supercalcolo per università e ricerca. "La tecnologia HPC ha la sorprendente capacità di allargare gli orizzonti dell'ingegneria, della ricerca medica, dell'esplorazione e di altri importanti ambiti che finora sono stati troppo onerosi e complessi da utilizzare e gestire efficacemente" ha affermato Bob Muglia, Senior Vice President Server and Tools Business di Microsoft. "Microsoft sta cercando di introdurre in modo significativo la tencologia HPC nel mercato, fornendo a dipartimenti e divisioni dell'industria e del settore pubblico I vantaggi in termini di costi, semplicità d'uso, e di sup-

ZIO BILL ADDIO!

Doveva arrivare anche il giorno in cui il buon vecchio Bill Gates abbandonava la Microsoft, azienda da lui creata quasi dal nulla, per dedicarsi alla beneficienza. E' quel giorno è previsto per l'anno 2008. A dichiararlo è stato lo stesso Bill Gates che ha annunciato di voler diminuire il suo impegno in Microsoft, riducendolo a quello di un consulente tecnico, per dedicarsi invece a tempo pieno alla "Bill e Melinda Gates Foundation", un'organizzazione con scopi di beneficienza fondata dallo stesso Bill Gates e piuttosto conosciuta in America. Gates ha spiegato di essere pienamente soddisfatto del proprio lavoro e di avere raggiunto traguardi tali sul piano economico e personale da sentirsi in dovere di restituire quanto la vita gli ha dato, a coloro che ne hanno più bisogno. A prendere il posto di Chief Software Architect che già fu di Bill Gates sarà Ray Ozzie. Craig Mundie che assumerà invece il ruolo di Chief Research and Strategy Officer. Nessuno scossone alla valutazione in borsa dell'azienda di Redmond si è verificato in seguito a questo pur importante annuncio.

porto ei partner tipici della piattaforma Windows Server. Il nostro obiettivo è far diventare la tecnologia HPC una risorsa diffusa – così facile da utilizzare proprio come sono le stampanti oggi”.
Certamente l’ingresso di un colosso come Microsoft in un mercato in cui fino a ieri Il leader indiscusso era Sun, apre nuovi scenari. Riuscirà la grande e potente Microsoft a convincere I centri di calcolo, le Università e I laboratori di eccellenza ad effettuare una migrazione delle loro piattaforme verso la tecnologia Windows Server? Certamente se l’operazione dovesse riuscire, la casa di Redmond assesterebbe un altro duro colpo alla concorrenza, confermandosi come società in grado di offrire tecnologia anche ad elevate prestazioni in qualunque settore dell’informatica moderna anche la dove le prestazioni costituiscono un requisito irrinunciabile.

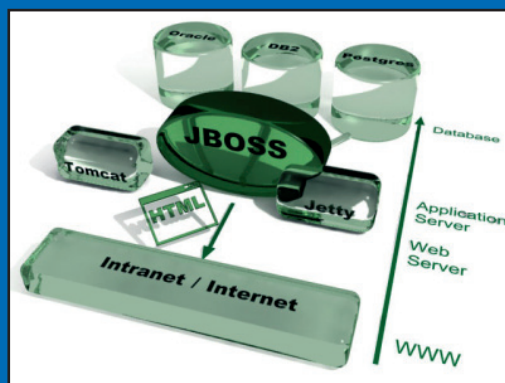
LE CREATIVE COMMONS SBARCANO SU OFFICE

3Sharp, un noto Solution Provider di Microsoft, realizzerà per conto dell’azienda di Redmond un tool gratuito che consentirà a chi lo desidererà di inserire le Creative Commons Licence in un documento realizzato con Microsoft Office con un solo click del mouse. Il tool sarà reso disponibile sul sito di Office Online.
La notizia è di quelle che scotta. Di fatto un’enorme quantità di documenti vengono oggi prodotti grazie alla suite di Microsoft e la possibilità di poter inserire facilmente il riferimento alla licenze Creative Commons rappresenta una svolta epocale nella diffusione della documentazione libera.
Lawrence Lessig, giurista e promotore delle Creative Commons si è dichiarato entusiasta di questa collaborazione con Microsoft che certamente porterà il logo delle CC su moltissima documentazione. Apprezzamenti sono arrivati anche dal Brasile, dove Gilberto Gill, ministro della cultura, intravede grandi possibilità da questa collaborazione fra imprese commerciali ed enti no profit. Il tool di 3Sharp sarà presentato proprio in Brasile durante l’iSummit o ovvero l’incontro che annualmente chiama a raccolta tutte le community ed I grandi esperti di “democrazia digitale” per discutere di come questo nostro mondo potrebbe evolvere. A dir la verità non sono in molti a credere alla bontà di Microsoft nel voler concedere a Lessig questo “calumet della pace”. Di fatto Lessig è testimone contro Microsoft nella causa che impegna l’Antitrust contro Microsoft già da molti anni.

CINQUE MILIONI DI OPENSOLARIS

È questo il traguardo raggiunto da OpenSolaris in un anno di vita. I risultati ottenti da Sun da quando ha deciso di dare vita alla comunità OpenSolaris scegliendo di utilizzare il modello di business OpenSource come strumento per fare breccia nel mercato sono eccezionali. Si contano circa 14.000 aderenti alla community e ben 5.000.000 di utenti registrati. Nessun sistema operativo prodotto da Sun aveva mai raggiunto traguardi simili. "Da quando abbiamo rilasciato Solaris nella comunità open source i nostri obiettivi di crescita annuale del business e della diffusione del sistema sono stati abbondantemente superati. Oltre l'85% delle società Fortune 500 utilizza Solaris 10 in sviluppo o in produzione", ha dichiarato Rich Green, Executive Vice President of Software di Sun Microsystems. "Questa crescita eccezionale è il risultato diretto della nostra strategia: investire per realizzare il sistema operativo più avanzato, adottare un modello di business all'insegna del Software come Servizio e far crescere la comunità degli sviluppatori Sun grazie al software gratuito e open source”. Ed è recente l'annuncio della disponibilità di una nuova release di Solaris 10, la 6/06.

AJAX CONQUISTA JBOSS

JBOSS è probabilmente l’application server più diffuso in ambito enterprise quando si tratta di servire aziende che necessitano di soluzioni solide e affidabili. Da poco JBoss è stato acquisito da RedHat ed a distanza di pochissimo tempo ecco arrivare Seam, un framework per lo sviluppo di applicazioni Web in tecnologia Java con una spiccata propensione ad Ajax. Il concetto è semplice: esporre una serie di interfacce, metodi, classi, proprietà, widget che orientino fortemente il programmatore a scrivere applicazioni che facciano uso della recente tecnologia. Lo scopo è quello di mantenere la solidità delle applicazioni, ma anche quello di fornire all’utente un’interfaccia molto user friendly più vicina a quelle delle comuni interfacce standalone che non a quelle che siamo abituati a vedere sul Web. Seam oltre che di Ajax farà uso anche delle nascenti Java Server Faces che rappresentano una tecnologia ormai matura per la separazione dei contenuti dalla logica di business. Con questo passo JBoss mostra una particolare attenzione a tutti gli ambiti della programmazione, garantendo non solo l’ormai proverbiale affidabilità ma anche una certa facilità nello sviluppo di interfacce che siano in linea con l’orientamente sempre più moderno e user friendly a cui il web sta tendendo.

DATABASE A PROVA DI CRASH

IMPARIAMO A PROGETTARE APPLICAZIONI CHE NON SMETTONO DI FUNZIONARE ANCHE IN CASO DI DANNI IRREPARABILI COME LA ROTTURA DI UN HARD DISK. ANALIZZEREMO ALCUNE TECNICHE DI FAILOVER INNOVATIVE, COME IL DB MIRRORING...

In questo articolo affronteremo un problema piuttosto complesso che non sempre può essere risolto con una soluzione univoca. Stiamo parlando del "failover". Cos'è un "failover". La rispo0sta è semplice, è un metodo per garantire che la nostra architettura software continui a funzionare anche in presenza di errori irreversibili. L'esempio sarà chiarificatore: la nostra azienda dispone di un database centralizzato e di un certo numero di client che si connettono a questo db per fare una serie di operazioni. Poiché l'amministratore di sistema è decisamente un tipo sfortunato, l'hard disk centrale su cui è installato il database si rompe completamente! A questo punto entrano in gioco le tecniche di failover che dovrebbero consentire all'azienda di continuare a lavorare nonostante si sia verificato un problema così grave.
Le possibili soluzioni sono:

1) L'amministratore ha fatto un backup dei dati e può ripristinare tutto su un secondo hard disk. Gli svantaggi di questa soluzione sono evidenti. Prima di tutto il backup dei dati non potrà essere stato eseguito in tempo reale. Perciò al successivo ripristino potremmo avere perso qualche dato e se siamo abbastanza sfortunati il numero di operazioni compiute dopo l'ultimo backup e prima del crash sarà elevatissimo. In secondo luogo l'azienda non potrà utilizzare il db fino a che l'amministratore non avrà ripristinato il tutto, e questo tipicamente comporta un certo tempo d'attesa.

2) L'amministratore è un tipo previdente e ha installato il database in un cluster. Ora diciamo subito che questa soluzione è probabilmente una delle più interessanti, ma presenta qualche svantaggio. Un cluster è sostanzialmente un "contenitore" di servizi trasparente ai suoi consumer. Facciamo subito un esempio per chiarirci ogni dubbio. Abbiamo installato SQL Server su tre macchine differenti: Server_A, Server_B, Server_C. Su una quarta macchina "MyCluster" abbiamo installato Windows Server 2003. Possiamo riunire Server_A, Server_B. Server_C sotto l'unica macchina MyCluster, che funzionerà come una sorta di contenitore virtuale. Ogni volta che a MyCluster arriverà una richiesta per usufruire dei servizi di database, il cluster sceglierà quale dei suoi nodi è attualmente più scarico e dirotterà ad esso la richiesta. Infine Server_A, Server_B e Server_C saranno mantenuti sempre sincronizzati, di modo che l'utente abbia sempre la sensazione di utilizzare un unico DB. Questa soluzione ha molti vantaggi, ma anche alcuni svantaggi. Prima di tutto l'intero sistema deve soddisfare un certo numero di requisiti. Sulla macchina che fa da cluster devono essere installati Windows 2003, Microsoft Cluster Services, SQL Server 2005. MCS richiede anche Active Directory, quindi la macchina in questione deve far parte di un dominio o esserne un controller. Bisogna dunque configurare MCS e di conseguenza SQL Server per lavorare all'interno del cluster. Tutto questo non comporta una difficoltà elevatissima, ma comunque rappresenta un innalzamento della complessità dell'intero sistema. Infine bisogna dire che il livello di glanuralità del Clustering è relativo all'intero server DB e non al singolo database. Tutte queste considerazioni, in questo contesto, vanno prese come un accenno al clustering, che è argomento decisamente complesso. Naturalmente lavorando in profondità scopriremmo che è possibile configurare il Cluster in modo molto efficiente. Ma non ne parleremo in questo articolo

3) L'amministratore è un tipo in gamba. Ha scoperto che in SQL Server 2005 è stata implementata una nuova funzionalità: il Database Mirroring. Questa tecnica è abbastanza semplice a livello logico. Si configurano tre macchine o almeno tre. Una prima macchina conterrà il Database primario su cui si lavora costantemente. Una seconda macchina conterrà un'esatta copia del primo aggiornata in tempo reale. Sarà appunto il mirror del database centrale. Se succede qualcosa al database primario immediatamente il secondario ne prenderà il posto. Il tempo di reazione è di

crica 3 secondi, e ovviamente non c'è perdita di dati, visto che i due server sono mantenuti sincronizzati in tempo reale costantemente. La terza macchina avrà funzioni di "Testimone". Questa macchina non è strettamente necessaria ai fini dell'allineamento dei dati, lo è alla fine della rilevazione del failover. Funziona un po' come una sorta di arbitro fra i due server e comunica ai due partecipanti lo stato in cui ciascuno dei due si trova. In questo articolo illustreremo appunto questa architettura.

TECNICHE DI BACKUP SEMPLICI

Prima di avventurarci nella descrizione della nostra soluzione, sarà utile descrivere qualche tecnica base, che ci consente di ottenere un primo livello di sicurezza. Inizieremo ovviamente, con il classico backup. Supponiamo dunque di avere un database chiamato "ioProgrammo_DB" su un'istanza di SQL Server presente sulla macchina 192.168.1.3 e di volerne effettuare il backup. In scenari di produzione è abbastanza improbabile che il sistemista lavori direttamente sul server, piuttosto lavorerà su una macchina client che contiene un'installazione di SQL Server Management Studio. E' esattamente il nostro scenario. La nostra macchina client sarà il 192.168.1.2. Utilizzeremo questa macchina per connetterci al primo server ed effettuare un backup del database ioprogrammo_db in esso contenuto. Dalla nostra macchina 192.168.1.2 in SQL Server Management Studio, in modalità SQLCmd diamo il seguente comando

```
:Connect tcp:192.168.1.3\SERVERA -U sa -P
                                          password
Backup database ioProgrammo_db to DISK =
          'C:\Backup\ioProgrammo.bak' with INIT
GO
```

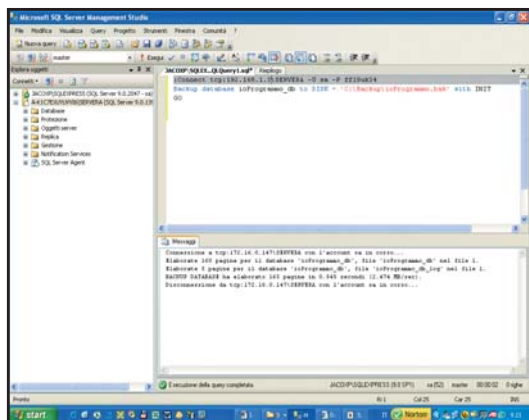



Fig. 1: Scenario di SQL Server Management Studio durante l'esecuzione del comando di Backup

LA MODALITÀ SQLCMD

SQL Server 2005, durante il setup, provvederà ad installare una serie di utility, alcune delle quali anche utilizzabili a linea di comando. Fra le tante una interessante è SQLCmd.exe che consente di connettersi a database remoti per eseguire su di essi operazioni direttamente da una Shell. L'utilizzo è semplice

```
SQLcmd -S tcp:indirizzohost\ServerA
                    -U sa -P password
```

A questo punto è possibile inserire comandi locali al server selezionato, oppure usare delle macro predefinite in SQLCmd. Ad esempio:

```
1> :serverlist
```

nel nostro caso restituirà

```
Server:
    A-K1C7EXUYUVVI6\SERVERA
    A-K1C7EXUYUVVI6\SERVERB
    ADOMANICO
    DBAMM
    INTRANET
    JACOXP
```

Ovvero l'elenco dei Server SQL presenti nella rete locale. In SQL Server managent studio è possibile abilitare una modalità ristretta di SQLCmd che ci consente di inviare comandi remoti ad una macchina direttamente dall'editor di Query. Per farlo è necessario abilitare il pulsantino SQLCmd come vi mostriamo nella figura seguente [immagine due.png]. In questa modalità i comandi supportati da SQLcmd sono i seguenti:

```
[:]go [count]
!! <command>
:exit(statement)
:Quit
:r <filename>
:setvar <var> <value>
:connect server[\instance] [-l
          login_timeout] [-U user [-P
                          password]]
:on error [ignore|exit]
:error <filename>|stderr|stdout
:out <filename>|stderr|stdout
```

Nel nostro articolo utilizziamo spesso il comando :connect che serve appunto a connettersi ad una macchina remota.

Il restore del database appena creato può avvenire con il seguente comando:

```
:Connect tcp:192.168.1.3\SERVERA -U sa -P
                                   password
RESTORE DATABASE ioprogrammo_db FROM DISK =
                  'C:\backup\ioprogrammo.bak'
WITH MOVE 'ioprogrammo_db' TO
             'C:\ServerA\ioprogrammo_db.mdf',
   MOVE 'ioprogrammo_db_log' TO
          'C:\ServerA\ioprogrammo_db_log.ldf',
   NORECOVERY
```

L'opzione MOVE fa si che i database in questione vengano "Spostati" nella directory specificata.

ROLLBACK E ROLLFORWARD

In ogni momento SQL Server mantiene un log delle transazioni effettuate. In condizioni di ripristino di un DB è possibile tentare di ripristinare l'ultima transazione ed in questo caso si parla di RollForward, oppure tentare di tornare indietro allo stato precedente all'ultima transazione eseguita, ed in questo caso si parla di RollBack

L'opzione NORECOVERY fa in modo che non venga eseguito il ROLLBACK delle transazioni ed abilita il conseguente ROLLFORWARD.
Utilizzando queste tecniche è possibile eseguire un backup periodico dei database. In casi estremi e non volendo interagire con lo scheduler di SQL Server 2005 si può anche utilizzare SQLCmd in congiunzione allo scheduler di sistema per eseguire backup periodici dei database.

DATABASE MIRRORING

Abbiamo già accennato all'architettura prevista dal database mirroring. Entriamo nel particolare e cerchiamo di capire esattamente qual è il flusso delle operazioni che viene eseguito in presenza di una configurazione come quella accennata

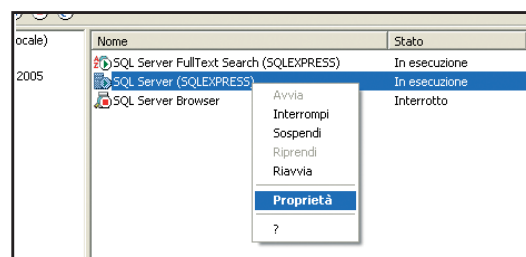
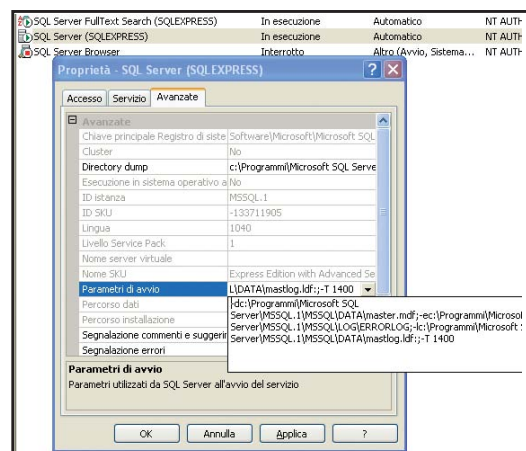
- Un client invia una richiesta di transazione al server principale di SQL Server 2005
- Il principale trascrive la richiesta di transazione nel file di log
- Il file di log viene inviato al server secondario o mirror
- Il server mirror esegue la transazione, valida il log e restituisce un OK al server principale
- Quando il server principale riceve l'OK esegue la transazione e invia un messaggio di conferma al client

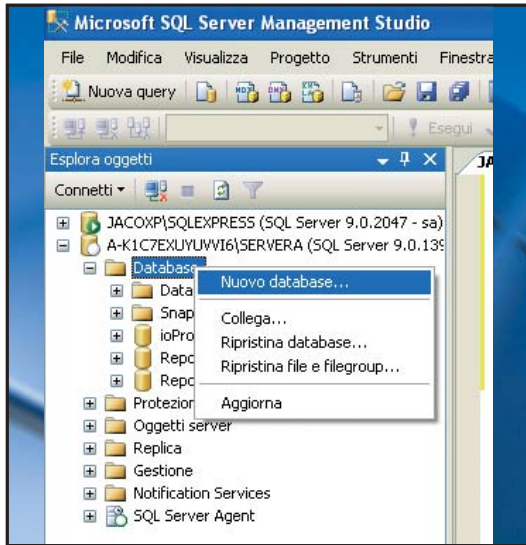
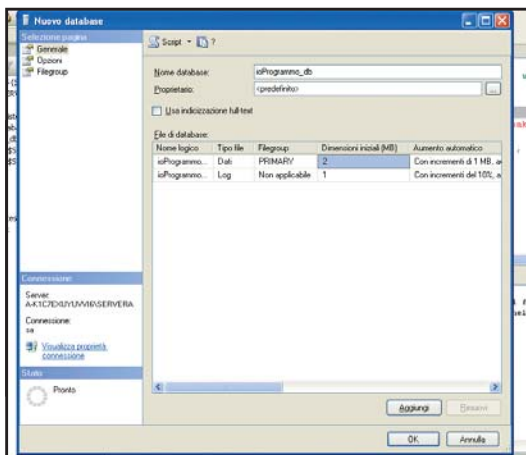
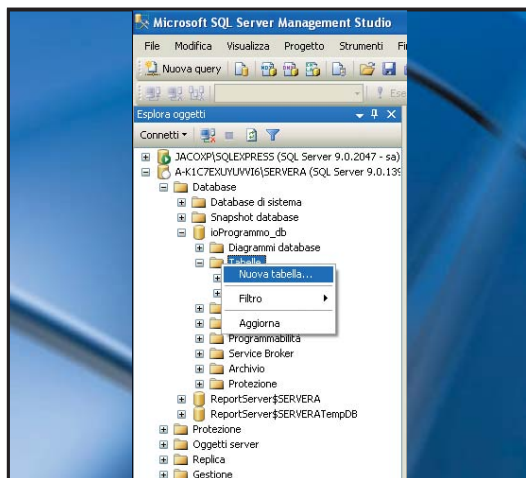
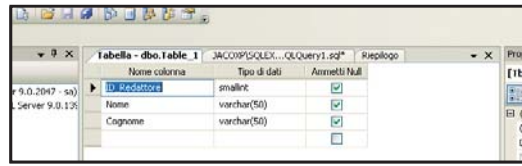
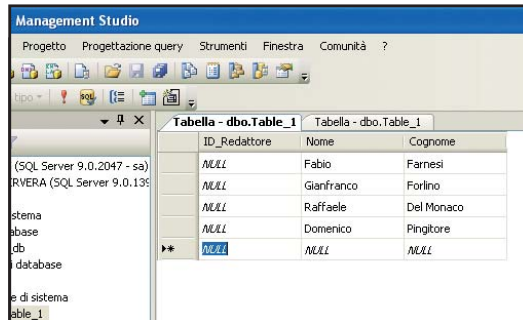
Fig. 2: ***Schema di funzionamento del mirroring in due condizioni tipiche***

Non abbiamo detto ancora niente a proposito del ruolo del server di controllo del mirroring. In effetti la presenza di questo componente è opzionale, tuttavia la presenza di "witness" o "testimone" rende automatico il failover.
Il server di controllo non interagisce nello scambio dati e non contiene nessun database. Il suo ruolo è esclusivamente quello di controllare che ad un dato momento esista un "Quorum" di server sufficienti a tenere in linea almeno un database come principale. Nella figura seguente viene illustrato il diagramma di funzionamento del mirroring in due tipici sceari di failover

DALLA TEORIA ALLA PRATICA

In condizioni standard il "Database mirroring" non è disponibile su Sql Server, deve essere reso disponibile abilitando il Flag di traccia 1400. Per farlo è necessario eseguire i seguenti passi

1) Aprite il "SQL Server Configuration Manager" e agite con il tasto destro del mouse sul nome che rappresenta l'istanza che desiderate configurare

2) Portatevi nella tabsheet "Avanzate" e aggiungete il parametro –T 1400

A questo punto riavviate il servizio
Per comodità creiamo un nuovo database ioprogrammo_db che contenga la tabella redazione.

Facciamolo direttamente da SQL Studio Manager, agendo tramite il pulsante destro del mouse su “Nuovo Database”

E attribuendogli un nome opportuno nella schermata che segue

Creiamo anche una tabella agendo con il pulsante destro del mouse su “Nuova Tabella”

E aggiungiamo qualche campo

Infine aggiungiamo qualche dato alla tabella

Verifichiamo anche che la protezione delle transazioni sia in modalità FULL

A questo punto eseguiamo il backup del database, come abbiamo imparato a fare tramite SQLCmd, oppure tramite SQL Management Studio, agendo con il pulsante destro del mouse su “Attvità/Backup”. Tramite SQLCmd il comando da impartire è il seguente

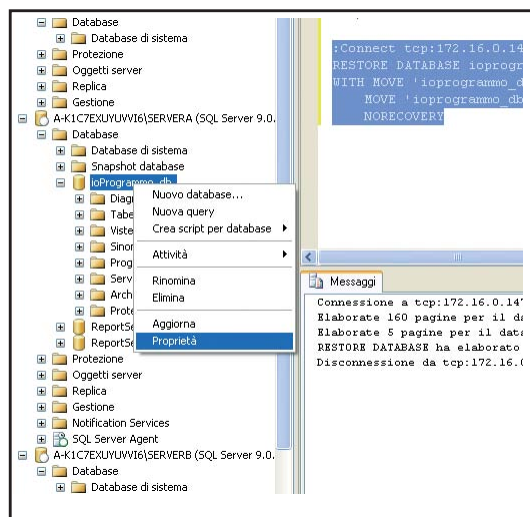
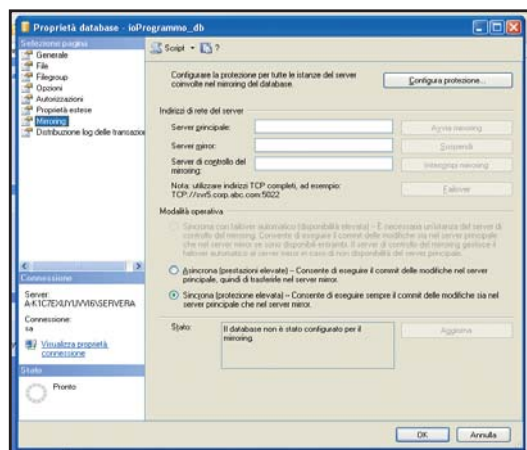
```
:Connect tcp:192.168.1.3\SERVERA -U sa –P password
Backup database ioProgrammo_db to DISK =
                    'C:\Backup\ioProgrammo.bak' with INIT
GO
```

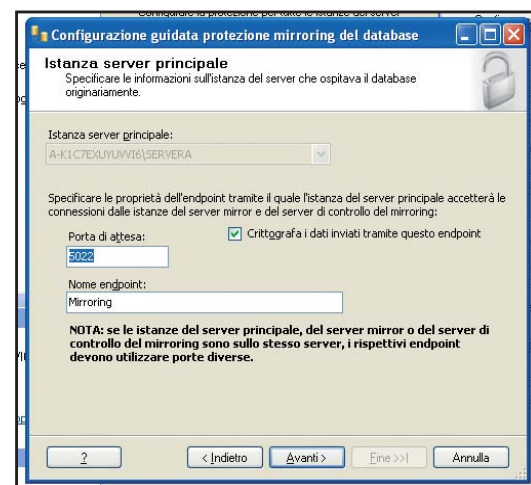
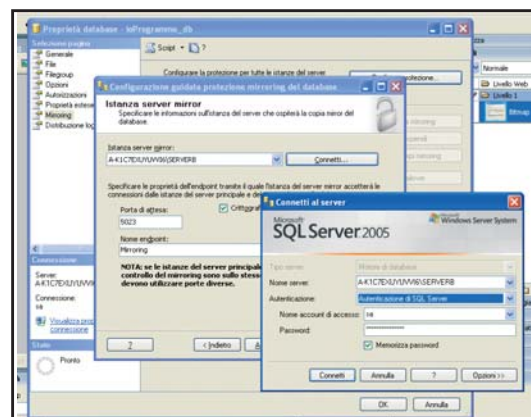
E provvediamo a ripristinare lo stesso database su una seconda istanza che sarà quella che utilizzermo come “Mirror”

```
:Connect tcp:192.168.1.3\SERVERB -U sa -P
                                        password
RESTORE DATABASE ioprogrammo_db FROM DISK =
                          'C:\backup\ioprogrammo_db'
WITH MOVE 'ioprogrammo_db' TO
                   'C:\ServerB\ioprogrammo_db.mdf',
    MOVE 'ioprogrammo_db_log' TO
                'C:\ServerB\ioprogrammo_db_log.ldf',
    NORECOVERY
```

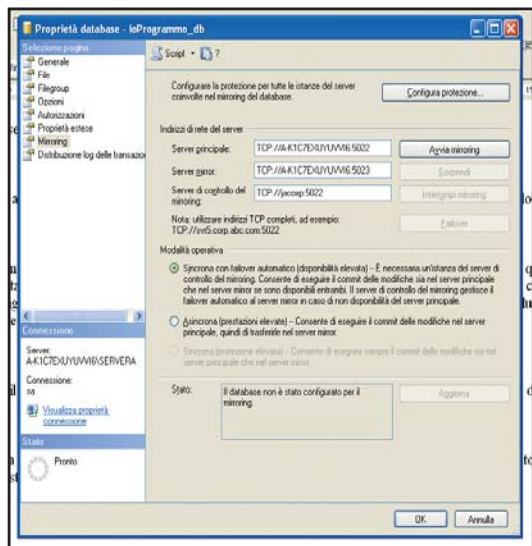
CONFIGURIAMO L’ARCHITETTURA PER IL MIRRORING

A questo punto configuriamo il mirroring. E’ possibile farlo anche a linea di comando, ma noi utilizzermo una comoda procedura guidata di SQL Management Studio. Posizioniamoci dun-

que sul server principale e agiamo con il tasto destro sul database di cui vogliamo configurare il mirrroring, scegliendo l'opzione proprietà dal menù contestuale

Nella finestra che segue scegliamo "Mirroring" e avviamo il wizard agendo sul tasto "Configura Protezione"

Rispondiamo "sì" alla prima opzione, dato che il nostro intento è proprio quello di configurare una sistema con failover automatico

Procediamo lasciando le impostazioni di default fino alla dialog box che ci chiede quale sarà il server principale. In questa finestra l'unica nota importante riguarda la porta di configurazione dell'EndPoint. Le comunicazioni fra i vari EndPoint che compongono il sistema di mirroring avverranno infatti tramite una specifica porta TCP. Il Nome dell'EndPoint può essere uno qualunque. Per EndPoint si intende il canale di comunicazione che verrà utilizzato per lo scambio dati.

Configuriamo poi il server di mirror, avendo cura di utilizzare il pulsantino connetti, per fornire le nostre credenziali di autenticazione

CONTINUARE IL RIPRISTINO IN CASO DI ERRORE

Se durante l'operazione di ripristino di un database dovesse verificarsi una condizione d'errore è ancora possibile recuperare un certo numero di dati. In particolare SQL Server 2005 consente l'operazione RESTORE CONTINUE_AFTER_ERROR. In questo caso viene tentato un ripristino di tutti i dati possibili non coinvolti nell'errore. Se si esegue un'operazione di CONTINUE_AFTER_ERROR, al termine del ripristino si potrebbero ottenere dei database inconsistenti, in tal caso è opportuno controllare la validità del database con il comando DBCC CHECKDB.

Infine con la stessa logica nella finestra successiva configuriamo il server di controllo del mirroring. Se tutto è andato a buon fine otterremo una finestra simile alla seguente, e non ci resterà che avviare il mirroring attraverso l'apposito pulsantino

Diamo anche un'occhiata agli script che avremmo dovuto utilizzare per creare la configurazione dei vari endpoint se non avessimo voluto farlo con il wizard, ma tramite comandi

```
:Connect tcp:192.168.1.3\SERVERA -U sa -P
                                        password
/****** Oggetto:  Endpoint [Mirroring]    Data
          script: 06/14/2006 14:41:54 ******/
CREATE ENDPOINT [Mirroring]
          AUTHORIZATION [sa]
          STATE=STARTED
          AS TCP (LISTENER_PORT = 5022,
                              LISTENER_IP = ALL)
          FOR DATA_MIRRORING (ROLE = PARTNER,
           AUTHENTICATION = WINDOWS NEGOTIATE
, ENCRYPTION = REQUIRED ALGORITHM RC4)
:Connect tcp:192.168.1.3\SERVERA -U sa -P
                                        password
/****** Oggetto:  Endpoint [Mirroring]    Data
          script: 06/14/2006 14:42:26 ******/
CREATE ENDPOINT [Mirroring]
          AUTHORIZATION [sa]
          STATE=STARTED
          AS TCP (LISTENER_PORT = 5023,
                              LISTENER_IP = ALL)
          FOR DATA_MIRRORING (ROLE = PARTNER,
           AUTHENTICATION = WINDOWS NEGOTIATE
, ENCRYPTION = REQUIRED ALGORITHM RC4)
:Connect tcp:192.168.1.2\WITNESS -U sa -P
                                        password
/****** Oggetto:  Endpoint [Mirroring]    Data
          script: 06/14/2006 14:43:14 ******/
CREATE ENDPOINT [Mirroring]
```

```
          AUTHORIZATION [sa]
          STATE=STARTED
          AS TCP (LISTENER_PORT = 5024,
                              LISTENER_IP = ALL)
          FOR DATA_MIRRORING (ROLE = WITNESS,
           AUTHENTICATION = WINDOWS NEGOTIATE
, ENCRYPTION = REQUIRED ALGORITHM RC4)
```

Notate che in questo scenario test abbiamo utilizzato due istanze su un'unica macchina e la terza su una macchina separata. Questo perché l'articolo ha fini didattici, in fase di produzione potete tranquillamente utilizzare macchine separate.

Notate anche che proprio perché le istanze erano sulla stessa macchina abbiamo usato porte separate per ciascuna istanza.

ANCORA SU ROLLBACK E ROLLFORWARD

Durante la fase di ripristino del database, con l'opzione rollforward, vi sarete sicuramente accorti che il database è completamente inutilizzabile. Molto probabilmente rimarrà in uno stato di perenne "Reovering". In un box all'interno di questo articolo abbiamo già spiegato che l'opzione Rollforward tenta di ripristinare il database interamente fino all'ultima transazione. Questo vuol dire che potrebbero esserci all'interno del db da recuperare, delle transazioni che non sono state ancora sottoposte a Commit. Per tale motivo il Database rimane in un perenne stato di recovering fino a che non viene sottoposto ad una seguente operazione di RollBack.

ATTIVARE IL PROTOCOLLO TCP/IP

Perché tutto funzioni correttamente dovete abilitare il protocollo TCP/IP nelle varie istanze dei server. Per farlo utilizzate il SQL Server Configuration Manager alla voce "Protocol"

SERVER EXPRESS O STANDARD

Il Mirroring non è supportato dalle versioni Express, quindi dovrete utilizzare versioni Standard, Enterprise o Developer. In teoria sarebbe possibile utilizzare la versione Express come Witness. Nella pratica abbiamo riscontrato non pochi problemi con questa soluzione. Per cui il nosto consiglio è quello di utilizzare sempre e comunque versioni superiori

TEST E VERIFICA DELLA CONFIGURAZIONE

Prima di tutto eseguiamo una verifica semplicissima. Diamo uno sguardo al nostro SQL Server Management Studio e selezionando il database ioProgrammo_db sul server principale noteremo che viene evidenziato come: Principal Synchronized. Allo stesso modo diamo uno sguardo al mirror e notiamo che viene etichettato come Mirror,Synchronized/ Restoring. Il server mirror non è accessibile neanche in lettura. Un secondo test è ovviamente facilmente eseguibile creando in Visual Basic 2005 una semplice applicazione per la gestione del database appena creato. Vi risparmio la procedura guidata per la creazione del database. Ciò che conta è che la stringa di connessione deve essere

```
Data Source=192.168.1.3\SERVERA;Failover
        Partner=192.168.1.3\SERVERB;Initial
        Catalog=ioProgrammo_db;Persist Security
        Info=True;User ID=sa;Password=password
```

Notate che abbamo settato un server di failover immettendo le coordinate del nostro server di mirroring.
Lanciamo l'applicazione e eseguiamo qualche modifica sul database aggiungendo o rimuovendo i dati.
Quando siamo ben soddisfatti stoppiamo l'instanza del server principal e magia delle magie noterete che tutto continua a funzionare normalmente, possiamo cancellare, modificare, inserire nuovi dati e l'applicazione non subirà nessuno scossone. Il server mirror in questo momento è stato promosso come principal. Riavviamo il principal e notiamo che ancora una volta i dati sono allineati. La condizione iniziale è stata ripristinata. Anche stoppando il witness tutto funziona normalmente.
L'unico problema che possiamo avere stoppando il witness è che in caso i failover la promozione del mirror a principal deve essere fatta manualmente.

IL COMANDO DBCC CHECKDB

Il comando in questione esegue una verifica approfondita dell'intero database per testarne la consistenza. In particolare, esegue un CHECK_ALLOC sul database per la verifica della corretta allocazione degli oggetti, un CHECK_TABLE per controllare la consistenza delle tabelle, controlla se ci sono messaggi nel service brocker e infine controlla la validità dei cataloghi.
Il comando DBCC CHEKDB può utilizzare molti parametri. Un elenco completo è disponibile all'indirizzo http://msdn2.microsoft.com/it-it/library/ms176064.aspx

RIPRISTINO DEL DATABASE MASTER

Quanto abbiamo fin qui detto è tutto bellissimo! E funziona alla perfezione. Ma che cosa succede se a danneggiarsi è il database master? Il Database Master, così come il model e il distribution non sono mirrorabili. Tutti sanno che il DB master contiene le informazioni sulla gestione degli utenti, sugli altri DB presenti e su ogni altra impostazione riguardante il funzionamento del server, quindi un suo mancato funzionamento provocherebbe danni irreparabili.
Ora, è anche vero che se abbiamo proceduto a mirrorare i DB che ci servono, anche il danneggiamento del master sul primario, sul secondario o sul Witness non provocherà gravi interruzioni di funzionamento, tuttavia illustriamo qualche procedura per ripristinare/ricostruire un db Master danneggiato.
Il primo passo, quello più semplice, è creare frequentemente un backup del master. Ormai abbiamo imparato come fare per compiere tale operazione, perciò tutto quello che dobbiamo fare è scrivere una nuova query nel modo seguente:

```
:Connect tcp:indirizzoip\istanza -U utente -P
        password
Backup database master to DISK =
        'C:\Backup\master.bak' with INIT
GO
```

In realtà avevamo già visto questo comando. Ve lo riproponiamo per sottolineare che è possibile effettuare un backup del master solo in modo "completo".
In tutti gli altri casi è possibile utilizzare anche altre modalità, ad esempio quella "parziale
A questo punto per ripristinare il Master è necessario

1) Avviare un'istanza del server in modalità utente singolo. Ovvero digitando da una shell il seguente comando:

```
sqlservr.exe - m -s
        <computer_name>$<instancename>
```

2) Ripristinare il master come abbiamo fatto in precedenza con gli altri Database utilizzando una query di questo tipo

```
USE master
GO
RESTORE DATABASE master
   FROM DISK = 'C:\backup\master.bak'
GO
```

In questo caso notate che non abbiamo utilizzato l'opzione NoRecovery.
Se tutto andrà a buon fine il master verrà ripristinato, con la conseguenza che tutti i DB presenti al momento del suo danneggiamento verranno ugualmente ripristinati.
Ovviamente il ripristino avverrà in maniera speculare ai dati salvati nell'ultimo backup. Se avete effettuato modifiche dopo l'ultimo backup queste non saranno ripristinate.
Il miglior consiglio che possiamo darvi è quello di effettuare un backup del master immediatamente dopo avere effettuato una modifica.

RICOSTRUZIONE DEL MASTER

Se siete particolarmente sfortunati, è possibile che in seguito al danneggiamento del master SQL Server decida di non ripartire. In questo caso l'unico tentativo possibile da effettuare è quello della ricostruzione del master danneggiato.
Prima di SQL Server 2005 esisteva un comando "Rebuildm" che ci avrebbe aiutato nelle nostre operazioni. Questo comando non è tuttavia più disponibile in questa versione, quindi bisogna ricorrere al setup.exe al quale dobbiamo fornire dei parametri per la sola ricostruzione del master. In particolare, dal supporto di installazione di SQL Server 2005 utilizziamo il seguente comando

```
setup.exe /qn INSTANCENAME=<InstanceName>
        REINSTALL=SQL_Engine REBUILDDATABASE=1
                    SAPWD=<NewStrongPassword>
```

Il parametro 'qn' nasconde le finestre di dialogo, mentre quello 'qb' consente di visualizzare finestre di dialogo e messaggi d'errore. In seguito alla ricostruzione del master verranno persi tutti gli eventuali Service Pack installati, quindi è opportuno ripetere un nuovo update.

CONCLUSIONI

Come detto in apertura di articolo, il clustering rimane probabilmente il modo migliore per mettersi al riparo da eventuali problemi, tuttavia il DB mirroring ha alcuni vantaggi che lo rendono particolarmente interessante negli scenari più consueti, non ultimo quello di non comportare l'installazione di software aggiuntivo e quello di rispondere ad una gestione sufficientemente semplice.
Nonostante abbiamo dedicato poche righe all'applicazione client che sfrutta il DB Mirroring, vi sarete già accorti che se queste tecniche vengono utilizzate in congiunzione al framework .NET 2.0 , risultano particolarmente efficaci. Per un programmatore, supportare il DB mirroring si riduce ad aggiungere un parametro alla stringa di connessione.
SQL Server 2005 supporta un numero elevatissimo di opzioni per il backup dei dati. Tuttavia il Backup semplice non mette al riparo da eventuali crash fisici o assenza di connessioni. In tutti questi casi ci vien incontro il DB Mirroring.
Nonostante gli elevati livelli di sicurezza di SQL Server 2005 e a prescindendo dalle tecniche di recovering è importante ricordare che il primo passo da compiere per essere certi che i nostri dati siano al sicuro è il semplice backup, costante ed eseguito ad intervalli di tempo regolari, anche e soprattutto del db Master. Un problema può sempre accadere, ma l'importante è farsi trovare preparati!

GLI STATI DEL DATABASE

In ogni momento un database può essere in uno dei seguenti stati:

***OnLine*: E' lo stato classico. Il Database è in linea e pronto per essere usato**

***Offline*: Il database è stato portato non in linea e non è disponibile. E' un'operazione che deve essere compiuta dall'utente. Normalmente questo tipo di condizione è opportuna quando si vogliono spostare dei file da una locazione ad un'altra. Appena terminata l'operazione si può riportare il database in modalità OnLine**

***Restoring*: Si sta ripristinando un database da un backup Recovering: E' in corso il processo i recupero del db.**

***Recovery Pending*: si è verificato qualche errore durante il processo di recupero. Questa condizione non si modifica automaticamente ma richiede un qualche intervento manuale da parte dell'utente**

Suspect: C'è un problema sul database, probabilmente è danneggiato. Il Database non è disponibile e bisogna tentare un azione di recupero

***Emergency*: Il Database viene posto in questa condizione dall'utente. In questo stato il database è in modalità utente singolo, può essere corretto o ripristinato ed è consentito l'accesso in modalità READ_ONLY, sola lettura, a coloro che hanno i privilegi di amministratori del database**

STAMPA DI PAGINE WEB CON CSS

TUTTE LE PRINCIPALI NOVITÀ INTRODOTTE DALLA SPECIFICA CSS 2 PER QUANTO RIGUARDA LA CREAZIONE DI FOGLI DI STILE SPECIFICI PER LA STAMPA. ORA NON CI SONO PROPRIO PIÙ SCUSE PER NON CONVERTIRSI AI CSS!

Ai primordi, il web era una semplice rete digitale costituita da documenti statici, il cui obiettivo era principalmente quello di fornire informazioni su una molteplicità di argomenti. In seguito Internet ha subìto una rapida espansione, crescendo vertiginosamente sotto l'incalzare di nuove e sempre più sofisticate esigenze: si è passati, infatti, dalle pagine web statiche a documenti in grado di interagire con server remoti per modificare dinamicamente i propri contenuti, fino a giungere alle moderne frontiere dell'e-commerce e dell'e-banking, che inaugurano una nuova era di servizi tecnologici alla portata di tutti. Questa rivoluzione, come è facile intuire, con il passare del tempo ha allargato sempre di più il divario tra programmatori e grafici, intendendo per i primi coloro che si occupano soprattutto delle logiche di funzionamento del sito, e per i secondi coloro che invece curano maggiormente l'organizzazione e la presentazione dei contenuti. È evidente che ciò ha comportato una sempre crescente specializzazione delle due figure, e una maggiore caratterizzazione delle stesse, a differenza di ciò che avveniva nel passato quando spesso e volentieri i due ruoli tendevano a confondersi tra di loro.

CSS: LI CONOSCIAMO?

Alla luce di quanto sopra riportato, è evidente che al giorno d'oggi la realizzazione di un sito web compatibile con la maggior parte dei browser esistenti e conforme con gli attuali e soprattutto futuri standard qualitativi è un lavoro che richiede una grande attenzione, non soltanto per gli aspetti legati alla programmazione dei contenuti, ma anche e soprattutto per quelli legati alla forma e alla presentazione delle informazioni. Prendiamo, ad esempio, i CSS (fogli di stile): credo che qualunque sviluppatore in ambito Internet sappia cosa sono, li sappia utilizzare più o meno bene all'interno di un sito web, e così via. Credo anche, tuttavìa, che ben difficilmente uno sviluppatore alle prime armi, che magari ha poca esperienza di lavoro in ambito Internet, riesca a cogliere appieno le grandi opportunità che essi possono offrirgli. L'errore in cui più comunemente cade il webmaster alle prime armi, infatti, è proprio quello di tendere facilmente a privilegiare il "cosa devo fare" rispetto al "come lo devo fare", abolendo pressoché totalmente la fase di progettazione iniziale e adottando un modus operandi di stampo decisamente pragmatico, efficace però solo in apparenza.
Il risultato sarà, con ogni probabilità, la realizzazione di un sito web che non rispetta i principali canoni della web-usability. Giova constatare, a questo riguardo, come il famoso detto di Machiavelli "il fine giustifica i mezzi", tanto apprezzato nel mondo informatico attuale da divenirne la principale causa del declino qualitativo, in questo ambito più che in altri ha ormai fatto il suo tempo, spodestato dalla moderna esigenza di riconoscere il giusto spazio anche alla progettazione e alla corretta organizzazione dei contenuti, piuttosto che al solo sviluppo delle logiche programmatiche sottostanti. Ma ben lungi dal voler fare una crociata contro i cultori dell'"informatica facile", visto che non è sicuramente questo il momento più opportuno per farlo, ritorniamo alla domanda posta dal titolo del presente paragrafo: quanto in profondità conosciamo i CSS ?
Per rispondere a questa domanda partiamo subito dall'analisi di un caso pratico, che a prima vista può apparire scontato, ma che nella realtà nasconde qualche piccola insidia: vogliamo stampare una pagina web senza includere nell'output il pulsante di stampa. Vogliamo anche applicare, nel caso del layout di stampa, una formattazione grafica dei contenuti diversa rispetto a quella adottata a

video. Lo studio di questo esempio affronterà un argomento annoso ed estremamente importante com'è, appunto, la gestione della stampa delle pagine web, e nello stesso tempo illustrerà le principali caratteristiche e il funzionamento di un attributo dei CSS tanto importante quanto sconosciuto e sottovalutato: il media.

STAMPA: UN PRIMO ESEMPIO

Javascript dispone della funzione *print*(), che consente di effettuare la stampa di un documento web. Tale funzione rappresenta un metodo di *Window*, gerarchicamente il più elevato degli oggetti Javascript, e può essere utilizzata soltanto per stampare un documento web intero. L'idea, a questo punto, potrebbe essere: lavoro con 2 frame; uno che contiene il pulsante, l'altro che contiene il documento web da stampare. Su pressione del pulsante (evento onclick) deve essere invocata una funzione che stampa il frame dei contenuti.

```
stampa_layer.html
<html>
<head>
<meta http-equiv="Content-Type"
                    content="text/html">
<title>Esempio di stampa con frame</title>
        <script>
        function stampa() {
                primo.focus(); //sposta il
                        cursore sul primo frame
                primo.print(); //stampa il primo
                                        frame
        }
        </script>
</head>
<frameset cols="50%,50%">
        <noframes>
        Il tuo browser non supporta i frame
        </noframes>
        <frame src="primo.html"name="primo">
        <frame
            src="secondo.html"name="secondo">
</frameset>
</html>
primo.html
<html>
<head>
<meta http-equiv="Content
                Type"content="text/html">
</head>
<body>
```

```
        <center>
                Questa &egrave; la pagina che
                        deve essere stampata:
                il pulsante invece non compare
                                in stampa
        </center>
</body>
</html>
secondo.html
<html>
<head>
        <meta http-equiv="Content-Type"
                        content="text/html">
</head>
<body>
        <input type=button onclick="javascript:
            parent.stampa();" value="stampa">
</body>
</html>
```

La soluzione appena illustrata sicuramente funziona, qualcuno potrà però obiettare sul fatto che l'utilizzo dei frame, a tutt'oggi, è decisamente sconsigliato nella maggior parte dei casi, vista la grande quantità di inconvenienti (ampiamente descritti nel riquadro a lato) che essi presentano. Il sistema migliore, infatti, è quello di sfruttare le grandi potenzialità dei CSS, che consentono di lavorare dinamicamente sui vari elementi costitutivi della pagina web organizzandoli in layer: per questi ultimi sono tra l'altro disponibili proprietà che consentono di specificare se un elemento è visibile o meno. L'idea, a questo punto, può essere quella di invocare, su pressione di un pulsante, una funzione Javascript che, prima di stampare la pagina, renda invisibile il layer con il pulsante stesso.

```
<html> <head>
<script language="JavaScript">
function stampa()
    {
area=document.getElementById
                        ('area_stampa').syl;
        area.visibility="hidden";
```

SAPEVI CHE...

**Negli esempi di questo articolo sono stati usati due metodi per non far comparire un layer in fase di stampa: display:none e visibility:hidden.
Ma qual è la differenza tra i due? In pratica, se si usa visibility:hidden il layer coinvolto non verrà mostrato, ma ne sarà mantenuto ugualmente l'ingombro (quindi, è come se il layer venisse soltanto reso invisibile pur continuando a esistere fisicamente). Con display:none, invece, il layer viene proprio rimosso dall'output della pagina, e quindi non occuperà neppure spazio.**

```
            self.print();  //stampa se stessa
    }
</script>
</head>
 <body> Pagina di prova: questa riga deve
                 comparire anche sulla stampa
 <div id="area_stampa"
      style="position:absolute;top:100;left:100;">
 <form>
 <input type="button" value="Stampa"
                onclick="javascript: stampa();">
 </form>
 </div>
 </body>
 </html>
```

Pur essendo una soluzione apprezzabile come base di partenza per la nostra analisi, tuttavìa è evidente che la funzione sopra riportata diverrebbe molto più complicata se volessimo gestire anche tutti i comportamenti e gli stili che ogni browser deve adottare in fase di stampa e successivamente alla stessa. Ma soprattutto il problema principale è che non siamo ancora riusciti a separare totalmente le logiche di gestione della stampa da quelle legate al normale layout a video (es. se stampo direttamente dal browser, la pagina includerà inesorabilmente anche il pulsante di stampa).
Abbiamo sicuramente imboccato la strada giusta, tuttavia siamo ancora abbastanza lontani dall'obiettivo: e pensare che il problema, inizialmente, sembrava così banale !
Ebbene, a riprova del fatto che, come diceva il noto protagonista di tanti romanzi di Arthur Conan Doyle Sherlock Holmes "spesso siamo portati a ricercare lontano da noi la soluzione di un problema mentre non ci accorgiamo che ne abbiamo una molto più semplice a portata di mano", ecco che tutte le nostre difficoltà svaniscono semplicemente dopo una

STAMPARE CON DUE PAGINE DISTINTE

Sfrutteremo il metodo innerHTML di un layer, la pagina da stampare sarà generata dinamicamente. Questo ci consente di evitare la creazione di due copie effettive della stessa pagina, diverse solo per pochi elementi, che renderebbe di difficile manutenzione la pagina stessa, con conseguenti possibili disallineamenti tra le due versioni. Agendo poi sul metodo print() della finestra creata, si avvierà la stampa del documento. Ma se si volesse stampare la pagina senza visualizzare la finestra ? È possibile, ma bisogna ricorrere ad artifici particolari, per esempio impostando degli attributi di posizionamento della finestra volutamente esagerati rispetto alle coordinate del video, che quindi non ne permettono la visualizzazione. Un sistema, questo, molto fantasioso, ma che sicuramente non rappresenta una soluzione pulita ed elegante come'è quella che fa uso, invece, dei fogli di stile.
Es. pratico di gestione della stampa con creazione di due versioni distinte della stessa pagina:

stampa.html

```
<html>
<head>
    <title>Esempio di
    stampa con 2 versioni distinte
    della stessa pagina</title>
    <link rel="stylesheet"
          href="video.css">
    <script>
    function
          stampa_popup() {
          var codice =
 "<html><head><title>Pagina di
          stampa</title>";
          // il
    documento di stampa deve
    essere linkato al foglio di stile
          usato per la stampa
          codice +=
      "<link rel=\"stylesheet\"
       href=\"stampa.css\">";
          codice +=
              "</head>";
          codice +=
              "<body>";
          codice +=
 document.getElementById('livello
   _da_stampare').innerHTML;
          codice +=
      "</body></html>";
          var fin =
 window.open("","stampa","width
   =500, height=400"); //apro la
                          finestra
fin.document.open();
fin.document.write(codice);
fin.document.close();
                 // fin viene a
    contenere l'oggetto-finestra
                           creato
       fin.print(); //invoco la
  funzione print()sulla finestra per
             chiamare la stampa
                 fin.close();
      //chiudo la finestra generata
   dinamicamente dopo la stampa
             }
       </script>
</head>
<body>
<div id="livello_da_stampare">
<b>Nel mezzo del cammin di
       nostra vita, mi ritrovai<br>
per una selva oscura, ch&egrave;
    la diritta via era smarrita.</b>
<br><br>
<cite>Dante Alighieri</cite>
</div>
<br><br>
<input type=button
             value="Stampa"
          onclick="javascript:
          stampa_popup();">
</body>
</html>
```

più attenta lettura delle caratteristiche di base della specifica CSS stessa.
Evidentemente non serve quindi ricercare soluzioni fantasiose e complicate, basta semplicemente approfondire meglio lo studio degli strumenti (nel nostro caso i fogli di stile appunto) che già fanno parte della nostra "cassetta degli attrezzi". Ma in che modo i CSS ci vengono in aiuto ?

CSS: L'ATTRIBUTO MEDIA

In precedenza abbiamo avuto modo di accennare all'esistenza di un attributo dei CSS, il media: è ora venuto il momento di analizzarlo.
Grazie a tale attributo siamo in grado di impostare un diverso foglio di stile a seconda del supporto sul quale intendiamo distribuire la nostra pagina: questa opportunità, la cui utilità è maggiormente comprensibile se intendiamo gestire il layout di stampa di un documento, sarà importante, più in generale, anche per quei webmaster, particolarmente lungimiranti e attenti all'evoluzione di Internet, che desiderano fin da subito rendere il proprio sito web compatibile con tutti i futuri mezzi di diffusione delle pagine (X)HTML. Nel futuro, infatti, i normali browser potranno essere affiancati da palmari, cellulari, altri dispositivi e, perché no?, anche browser vocali per i disabili; Tutte tecnologie, queste, dotate di caratteristiche molto differenti tra di loro in termini di gestione della memoria, ampiezza degli schermi, ecc…, al punto da richiedere, quindi, specifiche di formattazione degli stili assai differenti le une dalle altre. Ma ritornando alla stampa, è necessario ribadire come l'organizzazione dei contenuti predisposta per la visualizzazione a video non è immediatamente applicabile alla riproduzione su carta. La soluzione, adottata da alcuni webmaster, di realizzare due versioni distinte della stessa pagina (delle quali una, ovviamente, destinata alla stampa), è sicuramente molto valida e semplice da realizzare, ma presenta molteplici inconvenienti legati soprattutto ai tempi di sviluppo e alla difficoltà di tener conto, a volte, di minime differenze tra una versione e l'altra (es. spesso la versione stampabile differisce da quella normale soltanto per un pulsante o una barra dei menu). Ma tale tecnica costringe il webmaster anche a ricorrere ad artifici particolari, per ottenere determinati effetti di uso comune.

FACCIAMOLO CON I CSS

Vediamo ora di affrontare il problema interamente con i CSS: in questo caso non creiamo più versioni dello stesso documento, ma semplicemente applichiamo uno stile diverso a seconda del mezzo di visualizzazione che intendiamo gestire. L'unico vincolo al quale dobbiamo attenerci, se vogliamo sfruttare appieno la potenza dei CSS, è l'organizzazione del codice (X)HTML secondo una struttura semantica, che impone di separare accuratamente i contenuti dalle logiche di presentazione e visualizzazione (le quali devono essere gestite interamente dai fogli di stile). Innanzitutto ecco l'elenco dei valori possibili per l'attributo media, accompagnati dai dispositivi di visualizzazione a cui si riferiscono:

- **all**: è il valore di default valido per tutti i mezzi di diffusione della pagina;
- **aural**: browser a sintesi vocale;
- **braille**: supporti basati sull'uso del braille;
- **embossed**: stampanti braille;
- **handheld**: palmari e dispositivi portatili vari;
- **print**: stampa su carta;
- **projection**: è impiegato per proiezioni e presentazioni a tutto schermo;
- **screen**: visualizzazione su schermo;
- **tty**: dispositivi a carattere fisso;
- **tv**: web-tv.

L'attributo media può essere usato sìa con l'elemento LINK che con l'elemento STYLE:

```
<link rel="stylesheet" type="text/css"
        media="screen" href="foglio_video.css" >
<link rel="stylesheet" type="text/css"
```

STRUTTURA GRAFICA O STRUTTURA SEMANTICA ?

Nel testo si è fatto riferimento alla struttura semantica, che impone una rigida separazione tra i contenuti (gestiti con l'XHTML) e le logiche di presentazione e visualizzazione (specificate nei fogli di stile). L'opposto della struttura semantica è la struttura grafica, tipica di molti siti che usano le tabelle, per esempio, non tanto per organizzare in modo logico le informazioni, ma piuttosto come "griglia" per suddividere i vari contenuti. Allo stesso modo, per esempio, la struttura grafica gestisce spesso i menu tramite i contenitori generici HTML (i DIV), mentre a livello semantico è molto più rigoroso usare le liste. Per ulteriori informazioni concernenti la realizzazione di documenti "well-formed" in XHTML, consiglio di visitare il seguente validissimo link: http://www.w3schools.com, che contiene vari tutorial su tutto ciò che riguarda il web, con linguaggio semplice e accessibile. Unico inconveniente: è in inglese !!!

NOTA

COME UTILIZZARE I FILE DI ESEMPIO

Sul CD allegato sono disponibili i file di esempio, raccolti in tre cartelle. E' sufficiente salvare le cartelle sul disco rigido, quindi aprire con un browser la pagina index.html presente in ciascuna di esse e utilizzare i link presenti per effettuare le varie operazioni (mandare in stampa il documento, visionarne il codice sorgente, ecc…).

```
              media="print"href="foglio_stampa.css" >
oppure:
<style type="text/css">
@import url(foglio_video.css) screen;
@import url(foglio_stampa.css) print;
</style>
o ancora per i fogli di stile inline:
<style type="text/css">
@media screen {
   ...
   /* qui andranno inseriti gli stili riferiti alla
                              visualizzazione a video */
   ...
}
@media print {
   ...
   /* qui andranno inseriti gli stili riferiti alla
                                               stampa*/
   ...
}
</style>
```

All'interno dell'attributo media, che è opzionale, si possono inserire anche più valori, separati tra loro da virgole. Sarà lo user agent che dovrà visualizzare la pagina (es. il comune browser) a caricare il foglio di stile giusto, sempre che tale pagina sia strutturata in modo corretto. È bene a questo punto segnalare, tuttavia, che la tecnica di stampa delle pagine web tramite CSS che stiamo affrontando non funziona con i browser (come Netscape Navigator 4) che non sono in grado di interpretare i fogli di stile specifici per la stampa. Vediamo ora quali sono le principali differenze di formattazione tra la visualizzazione a video e la stampa su carta, gestibili direttamente tramite CSS: un modo pulito ed elegante che ci consente di evitare di ricorrere a "giochini" come quelli descritti nei paragrafi precedenti.

ELIMINAZIONE DI ELEMENTI DALLA STAMPA

Il problema che ci siamo posti inizialmente era proprio questo: come eliminare determinati elementi che, nella stampa, risultano antiestetici (come menu, pulsanti, link, ecc…) o poco professionali (es. banner pubblicitari, strisciorni vari, ecc…).
Per poter eliminare il pulsante in fase di stampa, basta definire nel foglio di stile relativo la seguente formattazione:

```
#pulsante {
   display: none;
}
```

COLORI

Ricordiamoci che la pagina su cui stampiamo, generalmente, è bianca, quindi sarà meglio impostare caratteri di colore scuro su sfondo bianco:

```
body {
   background: white;
   color: black;
}
#contenuto {
   background: transparent;
}
```

Sicuramente il colore scuro è la scelta migliore, in quanto consente un maggiore contrasto con lo sfondo, e quindi una maggiore leggibilità.

CARATTERI

Per quanto concerne il carattere da utilizzare, c'è da dire che sullo schermo la risoluzione dei pixel è limitata, pertanto in genere si usano caratteri semplici e senza particolari elaborazioni: meglio quelli senza grazie ("sans-serif") come Verdana o Helvetica. Le dimensioni dei caratteri, inoltre, sono in

PERCHÈ NON USARE IL FRAME?

I frame sono uno degli elementi più contestati dalla moderna web-usability (l'insieme delle regole che devono essere rispettate per realizzare siti efficienti e conformi ai migliori standard qualitativi). Ne sintetizzo qui di seguito i principali motivi:

- **alcuni browser non li gestiscono bene, per cui quando si va a inserire una pagina nei bookmark (segnalibri) si finisce per salvare la prima pagina del sito, e non quella effettivamente voluta;**
- **anche il semplice salvataggio della pagina è reso molto più difficile dal fatto che non è immediatamente visibile ciò che si sta realmente salvando;**
- **i frame complicano e appesantiscono molto la struttura della pagina, rendendone molto più difficoltosa la manutenzione;**
- **infine essi si comportano in modo imprevedibile in tante altre situazioni, rischiando talvolta di compromettere la normale navigabilità del sito.**

C'è da rimarcare, peraltro, che ormai anche i principali punti di forza dei frame (es. la barra di scorrimento che compare per scorrere i contenuti eccedenti le dimensioni del riquadro) sono supportati pienamente dai layer, i quali però non presentano tutti gli inconvenienti sopra elencati. Per ulteriori informazioni sulla corretta realizzazione di siti web, fornisco i seguenti link: www.usabile.it, www.sitichefunzionano.it, www.usability.gov, www.useit.com (questi ultimi due in lingua inglese).

genere espresse in pixel. Su carta invece la risoluzione disponibile è molto superiore, per cui è consigliabile usare caratteri con grazie ("serif") come il Times. La dimensione dei caratteri stampati può essere anche inferiore rispetto a quella adottata per il video, visto che la lettura su carta è meno faticosa: questo consente sicuramente di risparmiare inchiostro e fogli. Essa, poi, come spiegheremo meglio nel prossimo paragrafo, è in genere espressa in punti.

```
body {
  font: 10pt Times New Roman;
}
```

SCELTA DELLE UNITÀ DI MISURA

Dato che il video dei PC è basato sui pixel, l'unità di misura più adatta per la misurazione dei suoi elementi sarà sicuramente il pixel. Sulla carta tale unità di misura non ha certamente molto senso, mentre sarà più opportuno utilizzare i punti (come abbiamo visto, per i caratteri) e i millimetri (per tutto il resto). Ricordiamoci anche che il foglio più utilizzato per le stampe ha il formato standard A4: 210 x 297 millimetri. Utilizzando i millimetri, abbiamo un controllo molto più diretto sul risultato finale della nostra stampa:

```
p { text-indent: 15mm; }
```

COLLEGAMENTI

Come per la normale rappresentazione su schermo, anche le pagine stampate dovrebbero avere bene in evidenza i link. Basterà impostare per essi una formattazione particolare, privilegiando il grassetto e la sottolineatura. È possibile, inoltre, cambiare colore ai link (es. blu scuro). I link visitati dovrebbero anche avere lo stesso aspetto di quelli non ancora visitati.

```
a:link, a:visited {
  text-decoration: underline;
  font-weight:bold;
}
```

I fogli di stile consentono anche di visualizzare l'indirizzo del sito web di riferimento, accanto al testo del link, in modo da dare un'informazione in più sulla carta dove il testo da solo, altrimenti, non avrebbe alcuna utilità:

```
a:link:after, a:visited:after {
  content: " ("attr(href)") ";
}
```

Questo metodo presenta però due inconvenienti: intanto non funziona su browser che non interpretano al meglio i fogli di stile, come Internet Explorer 6 e Opera 6. Inoltre, l'indirizzo visualizzato dopo il nome del collegamento è l'attributo "href" del link stesso. Quindi, per i link esterni non ci sono problemi, in quanto l'attributo "href" coincide con l'URL, ma per i link interni al sito, che in genere vengono espressi in modo assoluto (es: /percorso _assoluto/pippo.html) o in modo relativo (es. ../../cartella/pagina.html), l'indicazione di tale attributo è in effetti di scarsa utilità. Ci vengono in aiuto, quindi, in attesa che la specifica CSS di livello 3 intervenga per correggere questo problema, le seguenti due classi, riferentisi rispettivamente ai link interni e a quelli esterni:

```
a.link-interno:after {
  content:" [http://www.sitoweb.com/"attr(href)
                                              "]"
}
a.link-esterno:after {
  content:" [" attr(href) "]"
}
```

CONCLUSIONI

I file di esempio relativi a quest'articolo sono disponibili direttamente nel CD allegato al presente numero.
I CSS rappresentano una risorsa importante a cui ogni programmatore dovrebbe affidarsi nella progettazione della propria applicazione. Vero è che sono nati per consentire la creazione di layout più raffinati, ma è anche vero che se usati con sapienza, i CSS possono rappresentare la soluzione a problemi molto comuni come ad esempio quello della stampa analizzato in questo articolo. L'uso corretto di tutte le tecnologie in ambito Web dai CSS ad Ajax a Javascript è l'unica strada che Internet ha per migliorare ancora l'impatto che le pagin e hanno sugli utenti Ricordiamoci che quando si parla di Web, il risulato finale in termine di Output è visibile soltanto all'interno del Browser, per questo motivo proprio i CSS devono rivestire un ruolo centrale nella progettazione di ogni Web Application

Enrico Viale

Enrico Viale è specializzato nello sviluppo di applicazioni sia web-oriented che desktop in ambiente Windows. Chi desidera contattarlo per chiarimenti riguardo all'articolo, o per qualsiasi altro motivo, può farlo all'indirizzo enrico.viale@gmail.com.

MYSQL, HIBERNATE ED ECLIPSE: IL TRIO RE

OGGI, GRAZIE A STRUMENTI OPEN SOURCE, È POSSIBILE IMPLEMENTARE SOLUZIONI SOFTWARE ESTREMAMENTE EFFICIENTI E FUNZIONALI IN TEMPI E COSTI NEMMENO IMMAGINABILI FINO A POCO TEMPO FA.

Solitamente quando si progetta o sviluppa un "software gestionale", sia che esso si occupi di gestione del magazzino o di contabilità o altro ancora, ci si deve sempre dividere in due. Infatti, una volta raccolti i requisiti da parte del cliente (in generale del futuro utilizzatore del software), bisogna farsi carico di due aspetti fondamentali che poi debbono essere fatti cooperare:

- la progettazione del database che tenga conto di tutti i dati necessari e delle loro relazioni
- la progettazione dell'applicazione che renda possibile all'utilizzatore stesso di realizzare in maniera agevole tutte le operazioni necessarie al funzionamento del sistema informativo (per capirci interfacce grafiche e gran parte della business logic).

Far interagire questi due aspetti non è così immediato e semplice come può apparire in un primo momento; gli ostacoli che possono presentarsi all'integrazione di questi due componenti possono essere molteplici; di seguito ne elenchiamo alcuni a cui tenteremo di dare una risposta.

DATABASE E APPLICAZIONI UTENTE

Il primo ostacolo riguarda le differenze di skill in possesso delle figure che si occupano di progettare i database rispetto a quelle in possesso di chi sviluppa applicazioni front-end (siano esse lato web, win32 etc...). Nel primo caso infatti si adotta ad esempio il paradigma entità relazione (solitamente indicato con l'acronimo E-R), mentre nel secondo sarà sicuramente presente, tra le tante possibili, una filosofia object oriented. Inoltre non bisogna dimenticare che, dal punto di vista implementativo, questi "due mondi" utilizzano linguaggi profondamente diversi: da un lato avremo un dialetto SQL, mentre dall'altro un linguaggio procedurale (tipo C) o OO (tipo C++,JAVA etc...). Sotto l'aspetto architetturale le comunicazioni tra il database e l'applicazione avvengono tramite degli appositi driver che si fanno carico di gestire la connessione al RDBMS, trasformare in eccezione gli errori generati dal database, inoltrare le query dall'applicazione al DB e restituire i risultati dell'interrogazione sotto forma di classe (si pensi ai DataSet in .NET o ResultSet in Java) e così via. È proprio in questa "terra di mezzo" che si rilevano le maggiori difficoltà: infatti, ad esempio, se chi ha implementato il lato front-end non è la stessa persona che ha progettato ed implementato il lato back-end può spesso capitare che il primo passi dei parametri che il DB non si aspetti. Per tutta risposta dal lato DB verrà generato un errore di difficile interpretazione.

PERSISTENZA DEGLI OGGETTI... MA NON SOLO

Per tutti i motivi sopra elencati è nata l'esigenza di trovare un modo che permettesse di utilizzare gli oggetti interfacciandosi al db in maniera del tutto trasparente.
Quando si parla di persistenza degli oggetti si possono intendere varie tecniche, anche diverse tra loro. In questo articolo per persistenza degli oggetti intenderemo un cosa molto semplice (almeno concettualmente parlando):
salvare e recuperare un oggetto in o da una tabella di un database.
In altre parole vogliamo instaurare una relazione biunivoca tra una tabella di un database ed una classe, o meglio, tra un record della tabella e l'istanza di una classe. In questo modo sarà possibile utilizzare il paradigma object oriented senza preoccuparci degli aspetti legati specificatamente al database. Il prodotto che rende possibile tutto ciò (e molto di più) è il framework open source Hibernate, sarà infatti questo ultimo a farsi carico degli aspetti discussi fino ad adesso. La figura 1 riporta l'architettura complessiva di una applicazione basata su Hibernate. Senza scendere troppo nei particolari notiamo proprio che Hibernate si pone, a livello logico, tra lo strato applicativo ed il driver JDBC.

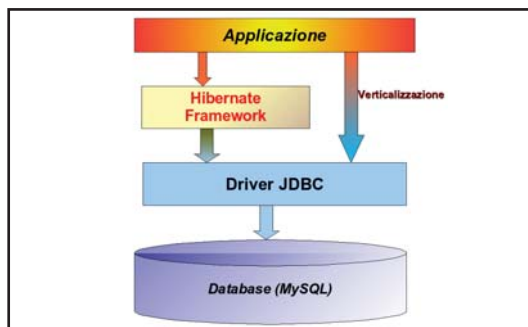



Fig. 1: *Architettura complessiva di una applicazione basata su Hibernate*

UN CASO REALE: GESTIAMO L'ORGANIZZAZIONE DELLA NOSTRA VIDEOTECA

Invece di dilungarci in considerazioni sulle funzionalità di questo potente framework (che sono numerosissime), vedremo come utilizzarlo da subito anche grazie all'aiuto di un validissimo plugin per Eclipse. Ad ogni modo, oltre all'abbondante documentazione bibliografica presente su Internet, per chi volesse approfondire ulteriormente alcuni aspetti di configurazione ed architetturali, l'articolo pubblicato sul numero di maggio sul suo “fratello” Nhibernate può sicuramente rappresentare un valido punto di riferimento. Tornando a noi, definiamo come prima cosa il problema che andremo ad affrontare, simuliamo cioè la fase che viene definita come “raccolta dei requisiti”:

La base di dati deve gestire l'elenco dei video in possesso dell'utente. Ogni video è memorizzato su uno o più supporti (nel caso di copie di backup) che debbono essere identificati da un numero univoco. Ogni video è identificato da un titolo e, per ciascuno di esso, possono essere specificate ulteriori informazioni come la data di uscita, il genere, il regista ed il cast di interpreti. Sia per i registi sia per gli interpreti è necessario specificare nome e cognome ed eventualmente la loro data di nascita. Per quanto riguarda il genere del filmato esso è identificato da un proprio nome (comico, drammatico, video clip, etc...). Questa non è la sede per discutere di progettazione E-R, quindi puntualizzeremo solo alcuni aspetti del nostro database, rimando quindi allo script in allegato alla rivista.

I vincoli esposti nei requisiti ci costringo a definire una tabella “*Interpretazione*” che ci permetta di risolvere la relazione molti a molti che sussiste tra la tabella Film e la tabella Attore.

Ecco infatti la struttura della tabella Film e quella di Interpretazione:

```
CREATE TABLE `FILM` (
  `ID` int(11) NOT NULL auto_increment,
  `TITOLO` varchar(50) NOT NULL default '',
  `REGIA_ID` int default NULL,
  `REGIA_COGNOME` varchar(20) default NULL,
  `GENERE` varchar(20) default NULL,
  `USCITA` date default NULL,
  PRIMARY KEY (`ID`),
  KEY `REGIA_ID` (`REGIA_ID`),
  KEY `ID` (`ID`),
  KEY `R_9` (`GENERE`),
  CONSTRAINT `direttoDa` FOREIGN KEY
          (`REGIA_ID`) REFERENCES `REGISTA` (`ID`),
  CONSTRAINT `R_9` FOREIGN KEY (`GENERE`)
                REFERENCES `GENERE` (`NOME`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `INTERPRETAZIONE` (
  `ATTORE_ID` int(11) NOT NULL,
  `FILM` int(11) NOT NULL default '0',
  PRIMARY KEY (`ATTORE_ID`,`FILM`),
  KEY `ATTORE` (`ATTORE_ID`),
  KEY `R_8` (`FILM`),
  CONSTRAINT `R_7` FOREIGN KEY (`ATTORE_ID`)
                REFERENCES `ATTORE` (`ID`),
  CONSTRAINT `R_8` FOREIGN KEY (`FILM`)
                REFERENCES `FILM` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Da notare inoltre come l'engine scelto sia InnoDB in modo tale che siano supportate anche le transazioni.

MENO MALE CHE C'È ECLIPSE!

Una volta creato il database dovremo occuparci di tutta la configurazione necessaria per interfacciare Hibrnate a quest'ultimo. Per chi di voi non avesse avuto nessuna esperienza in tal senso vi posso assicurare che si tratta di un'operazione alquanto tediosa, soprattutto per database di medio-alta complessità, che molte volte scoraggia lo sviluppatore ad utilizzare il framework stesso. Per nostra fortuna gli sviluppatori di Hibernate hanno creato un plugin per eclipse che ci sgrava da questi compiti automatizzando molti degli aspetti più meccanici. Prima di tutto, oltre al framework in questione, procuriamoci tale plugin (hibernate.org/255.html) ed installiamolo su Eclipse. A questo punto creiamo un nuovo Java Project ed assicuriamoci di referenziare tutte le librerie necessarie a Hibernate (per intenderci i JAR presenti nella sotto directory lib di hibernate più lo stesso hibernate.jar) e che i nostri sorgenti abbiano come directory principale src.

La prima cosa da fare è creare il file hibernate.cfg.xml che conterrà le informazioni generali per connettersi al database. I passi da seguire sono i seguenti:

SVILUPPO TRADIZIONALE

Come si vede dalla figura 1 l'utilizzo di Hibernate non preclude comunque la possibilità di continuare a sviluppare il codice interfacciandosi direttamente al JDBC. Questa precisazione è molto importante per tutti coloro che hanno applicazioni che utilizzano direttamente il database e vogliono affiancare una nuova soluzione basata su hibernate.

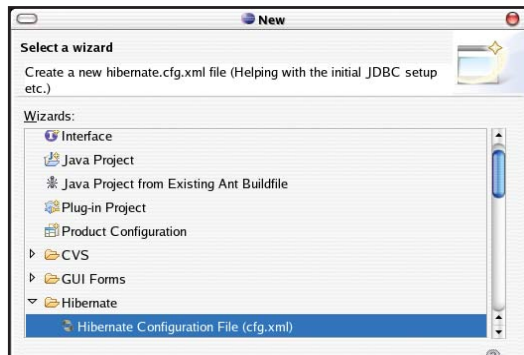
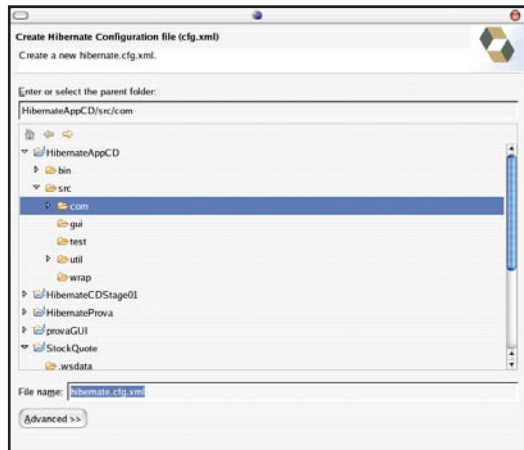
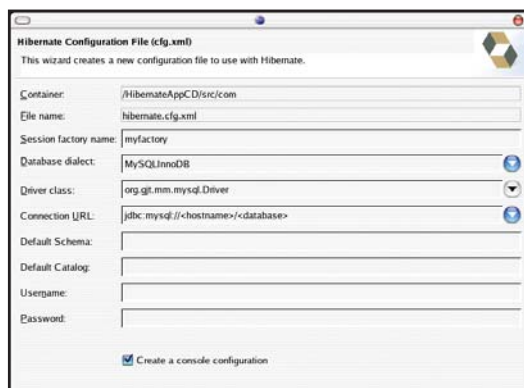
COSA È LA PERSISTENZA?

Per persistenza degli oggetti si può intendere genericamente tutte quelle tecniche che permettono di memorizzare lo stato degli oggetti su un qualsiasi supporto. La serializzazione ad esempio ci permette di salvare un oggetto su un file binario,In questo articolo quando parliamo di persistenza intendiamo sempre il salvataggio di un oggetto in un database.

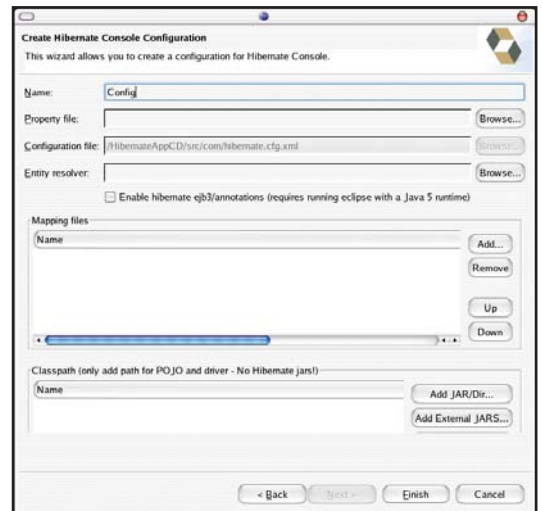
INSTALLARE UN PLUGIN IN ECLIPSE

Per installare un plugin basta decomprimere lo zip relativo nella root di installazione di Eclipse.

DRIVER JDBC IN HIBERNATE

Hibernate fornisce un proprio JDBC per connettersi a MySQL, nella fattispecie *org.gjt.mm.mysql.Driver.* Quello che mi sento di consigliare è di scaricare il JDBC fornito sul sito della stessa MySQL e di cambiare la relativa riga di configurazione con *com.mysql.jdbc.Driver.*

Scegliere File -> New -> Other dal menù di Eclipse, selezionare Hibernate Configuration File e premere Next

Selezionare la cartella src del vostro progetto e cliccare su Next.

Scegliere un nome per la session-factory, selezionare il tipo di database che si vuole utilizzare ed il relativo JDBC, e gli ulteriori parametri necessari (connectionURL, username, password etc...). Infine spuntate la voce Create a Console Configuration e premete Next

Come ultima operazione diamo un nome alla console.

Ora, all'interno della cartella src, dovremo avere il file XML hibernate.cfg.xml. Cliccando due volte si aprire un editor XML dedicato; il contenuto del file in questione dovrebbe essere questo:

```
<session-factory name="wrapFactory">
    <property name="hibernate.connection
     .driver_class">org.gjt.mm.mysql.Driver</property>
    <property name="hibernate.connection.
                    password">pwd</property>
    <property name="hibernate.connection
    .url">jdbc:mysql://localhost/cd_stage01</property>
    <property name="hibernate.connection.
                   username">user</property>
    <property name="hibernate.dialect">org.
      hibernate.dialect.MySQLInnoDBDialect</property>

...
```

A questo punto aggiungiamo due ulteriori property:

```
<property name="hibernate.connection.zeroDate
            TimeBehavior">convertToNull</property>

<property name="hibernate.current_session_
                  context_class">thread</property>
```

La prima ci assicura che una data non assegnata, che in database compare come 00-00-0000 00:00, verrà rimappata in Java come un oggetto null, mentre la seconda ci assicura che la sessione di Hibernate sia un thread.

ED ORA IL REVERSE ENGINEERING!

Quello che ora dovremmo fare, visto che siamo partiti dalla progettazione del database, è definire delle classi che mappino le tabelle create in database. In particolare per Hibernate bisogna seguire lo standard JavaBeans. Per scendere subito nelle considerazioni concrete diciamo quindi che ogni classe mappa una tabella dichiarando i campi di quest'ultima come variabili private ed esponendo i relativi metodi pubblici set/get. Facendo un esempio concreto sulla tabella CD del nostro DB

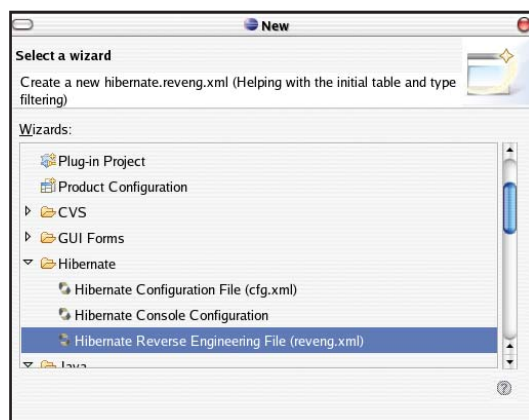
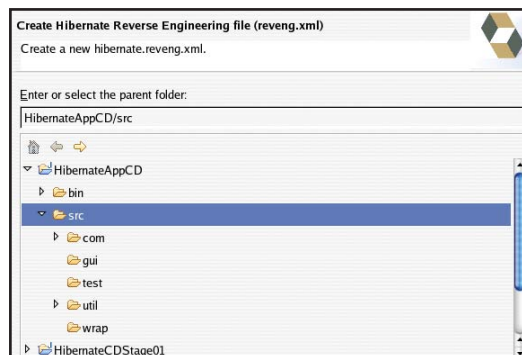
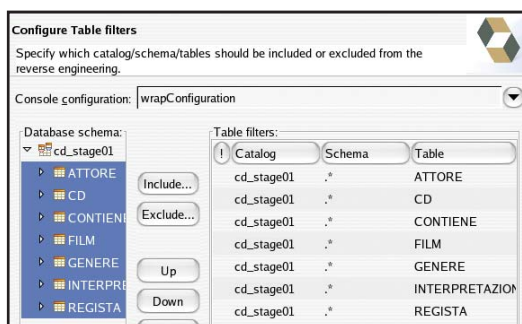
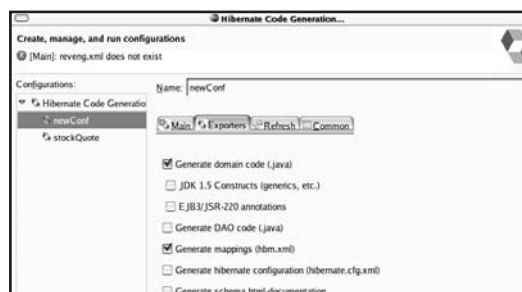
abbiano:

```
CREATE TABLE `CD` (
 `NUMERO` int(11) NOT NULL AUTO_INCREMENT,
 PRIMARY KEY (`NUMERO`),
 KEY `NUMERO` (`NUMERO`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

A questa tabella corrisponderà la seguente classe:

```
public class Cd {
  // Fields
   private int numero;
  /** default constructor */
  public Cd() {
  }
           /** minimal constructor */
  public Cd(int numero) {
     this.numero = numero;
  }
   // Property accessors
  public int getNumero() {
     return this.numero;
  }
  public void setNumero(int numero) {
     this.numero = numero;
  }
```

In realtà vedremo che sarà conveniente avere dei SET che tengano conto anche delle relazioni tra le classi. Per ora però fermiamoci un attimo e riflettiamo: in fondo tutte queste operazioni sono meccaniche e quindi facilmente automatizzabili! Anche in questo caso il plugin messo a disposizione da Hibernate ci risparmia un sacco di fatica, basta seguire i passi sotto riportati:

1 Selezionare New -> Other -> Hibernate -> Reverse Engineering Configuration File e cliccare su Next

2 Selezionare la directory dove sarà posto il file di reverse engineering

3 Selezionare la console configuration e cliccare sul bottone Refresh in modo tale da connettersi al database. A questo punto selezionare tutte le tabelle e premere il bottone Include

4 Dopo aver cliccato sull'icona di Hibernate selezionate Hibernate Code Generation. Sul tab Exporters selezionate generate domain code e generate mappings come mostrato in figura, infine cliccate su run.

Una volta completati tutti passi descritti sopra dovreste avere un package contenente tutte le classi che mappano le vostre tabelle e le relative relazioni fra esse. Inoltre vengo anche generati i file XML che definiscono quale campo di quale tabella debba essere mappato sull'attributo di quale classe. Ad esempio abbiano per la classe Cd il file Cd.hbm.xml:

```
<hibernate-mapping>
  <class name="wrap.Cd" table="CD"
                            catalog="cd_stage01">
    <id name="numero" type="int">
       <column name="NUMERO" />
       <generator class="assigned" />
```

Andrea Galeazzi
Laureato in ingegneria elettronica presso l'università Politecnica delle Marche, lavora presso un gruppo nazionale leader nel campo delle tecnologie e dei servizi IT. Nei limiti della disponibilità di tempo risponde all'indirizzo andreagaleazzi@fsfe.org

```
        </id>
        <set name="contienes" inverse="true">
            <key>
                <column name="CD" not-null="true" />
            </key>
            <one-to-many class="wrap.Contiene" />
        </set>
    </class>
</hibernate-mapping>
```

Ora l'unica cosa che c'è rimasta da fare è aggiungere al file di configurazione hibernate.cfg.xml tutti gli XML di mappaggio che il nostro tool ha generato. Nel nostro caso ad esempio basterà aggiungere:

```
<hibernate-configuration>
        <session-factory name="wrapFactory">

          <mapping resource="wrap/Cd.hbm.xml"/>
    <mapping resource="wrap/Attore.hbm.xml"/>
    <mapping resource="wrap/
                              Contiene.hbm.xml"/>
    <mapping resource="wrap/Film.hbm.xml"/>
    <mapping resource="wrap/Genere.hbm.xml"/>
    <mapping resource="wrap/
                         Interpretazione.hbm.xml"/>
    <mapping resource="wrap/Regista.hbm.xml"/>
            .............
```

Come avrete potuto sicuramente notare è necessario specificare il percorso completo delle risorse XML (wrap è infatti il package dove sono contenute).

ED ORA GODIAMOCI UN PO' DI CODICE IN STILE HIBERNATE

Siamo quindi ormai pronti per fare le nostre prime esperienze con questo framework. Partiamo ad esempio da una procedura banale: inseriamo un numero di CD nuovo in maniera transazionale (visto che come engine abbiano scelto InnoDB):

```
Session session =  HibernateUtil.getSessionFactory().
                                  getCurrentSession();
session.beginTransaction();
        Cd cd = new Cd();
        cd.setNumero(1);
        try {
session.save(cd);
        session.get Transaction().commit();
                } catch (HibernateException e) {
                // TODO Auto-generated  catch block
                        session.get
                              Transaction().rollback();
                System.out.println(e.
                                       getMessage());
        }}
```

In pratica come vedete non bisogna far altro che creare la vostra classe e salvarle tramite una sessione Hibernate. Ricordate inoltre che i salvataggi o gli update vengo effettivamete persistiti sul database solo quando viene invocato il metodo commit della sessione in questione. La classe HibernateUtil è una classe che ho implementato per rendere più agevole e rapido lavorare con le sessioni:

```
Public class HibernateUtil {
Private static final SessionFactory  sessionFactory;
        static{
                sessionFactory = new
        Configuration().configure().buildSessionFactory();

        }
        public static SessionFactory getSession Factory() {
                return sessionFactory;
        }
}
```

Per concludere vediamo come recuperare i CD che abbiano salvato sul database:

```
public static Cd[] getAll() throws HibernateException {
                Session s = HibernateUtil.
            getSessionFactory().getCurrentSession();
                try {
                        s.beginTransaction();
                        List res = s.create
                        Query("from wrap.Cd").list();
                        Cd[] out = new
                                  Cd[res.size()];
                out=(Cd[])res.toArray(out);
                        s.close();
                        return out;

                } catch (HibernateException e) {
                        catch block
                        throw e;
                }
        }
```

CONCLUSIONI

Come avrete notato da quest'ultimo stralcio di codice, Hibernate non fa uso del linguaggio SQL ma di qualcosa che gli assomiglia molto. Su questo aspetto torneremo nel prossimo articolo, per ora abbiano però già capito come, grazie a questo plugin, sia facile avere il framework configurato e pronto per l'uso. Inoltre prossimamente potremmo apprezzare come Hibernate ci agevoli anche nel progettare sistemi di GUI molto accattivanti.

Andrea Galeazzi

QUALE DATABASE?

L'RDBMS che si è scelto per questo articolo è il famoso MySQL, anche se in realtà utilizzando Hibernate potete scegliere il database che più preferite purché siate in possesso del relativo JDBC. In allegato alla rivista troverete un file script (create_db.sql) che vi creerà automaticamente il database su cui stiamo lavorando. Nel caso in cui anche voi abbiate deciso di avvalervi di MySQL i passi da seguire per creare il db sono semplicissimi:

1- Collegarsi al RDBMS

```
mysql -u <utente> -p
```

2- Creare il DB

```
create database
        <nomedatabase>
```

3- Uscire dal terminali

```
exit
```

4- Lanciare lo script

```
mysql -u <utente> -p
  <nomedatabase> <
        create_db.sql
```

DA SQL SERVER AL WEB IN UN LAMPO

LA NUOVA VERSIONE DEL DATABASE DI MICROSOFT, OFFRE UN SUPPORTO MIGLIORATO AD XML. VEDIAMO COME OTTENERE I DATI DIRETTAMENTE IN QUESTO FORMATO E COME TRASFORMARLI GRAZIE AD XSLT.

Microsoft SQL Server, a partire dalla versione 2000 ancor di più con la versione 2005, ha introdotto un supporto a XML veramente notevole.
XML viene utilizzato in Sql Server in diversi modi, in questo articolo, però, focalizzeremo la nostra attenzione al recupero dei dati.
Di cosa si tratta? In pratica della possibilità di estrarre i dati con una normale query per averli disponibili direttamente in formato XML.
A cosa serve? Pensiamo ad uno scenario di una Web Application dove abbiamo l'esigenza di estrarre i dati dal database per poi trasformarli in HTML per mostrarli in una pagina web.
Il metodo classico di fronte a questa esigenza, in ASP.NET, è solitamente quello di utilizzare il controllo DataGrid con l'associazione ai dati, oppure quello di costruire manualmente il codice HTML aprendo la connessione, scorrendo i dati e scrivendoli nell'output.
Entrambi questi sistemi presentano dei difetti:

- il primo approccio, con il controllo DataGrid, è troppo limitativo poiché il controllo sull'output (la tabella HTML che ne risulta) è, per forza di cose, abbastanza difficile; formattare tabelle molto complesse, magari con un po' di codice javascript o icone, è certo possibile ma, alla fine, risulta più complicato che scrivere direttamente il codice HTML
- il secondo metodo, quello preferito dai programmatori che vengono dalla scuola dell'ASP classico o del PHP, offre sicuramente più libertà ma ha il grave difetto di mischiare i dati con la logica di presentazione, con l'effetto che, successivamente, per cambiare anche solo il carattere del testo di una cella occorre scorrersi decine e decine di righe di codice.

Avere invece i dati in formato XML ci consente di trasformarli direttamente in HTML utilizzando XSLT, il linguaggio per la trasformazione di un sorgente XML beneficiando sia della separazione tra dati e contenuto che di un completo controllo sull'output da creare.
Per entrare un po' più sul concreto proponiamoci un obbiettivo preciso. Costruiremo una mini applicazione composta da due pagine web : nella prima mostriamo gli ordini ricevuti in una tabella, cliccando sull'ordine si aprirà nella seconda pagina la fattura per il cliente.

Conoscenze richieste
conoscenza di base di VB.NET, SQL e XSL

Software
Visual Studio 2005 anche in versione Express, Microsoft SQL Server anche in versione Express

Impegno

Tempo di realizzazione

ESTRARRE I DATI

Per ottenere i dati in XML con una query, in SQL Server, è sufficiente utilizzare una clausola, FOR XML, in coda al testo della query.
Prendiamo, ad esempio, la query sugli ordini :

```
SELECT TOP 10
OrderID,Orders.CustomerID,OrderDate,
                    RequiredDate,ShippedDate,ShipName,
                                          ShipAddress,
ShipCity,ShipPostalCode,ShipCountry,
CompanyName,Address,City,Country,PostalCode
from Orders
inner join Customers
ON Orders.CustomerID=Customers.CustomerID
```

Results | Messages

	OrderID	CustomerID	OrderDate	RequiredDate	ShippedDate
1	11077	RATTC	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL
2	11076	BONAP	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL
3	11075	RICSU	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL
4	11074	SIMOB	1998-05-06 00:00:00.000	1998-06-03 00:00:00.000	NULL
5	11073	PERIC	1998-05-05 00:00:00.000	1998-06-02 00:00:00.000	NULL
6	11072	ERNSH	1998-05-05 00:00:00.000	1998-06-02 00:00:00.000	NULL
7	11071	LILAS	1998-05-05 00:00:00.000	1998-06-02 00:00:00.000	NULL
8	11070	LEHMS	1998-05-05 00:00:00.000	1998-06-02 00:00:00.000	NULL
9	11069	TORTU	1998-05-04 00:00:00.000	1998-06-01 00:00:00.000	1998-05-06 00:00:00.000
10	11068	QUEEN	1998-05-04 00:00:00.000	1998-06-01 00:00:00.000	NULL

Fig. 1: ***esecuzione query standard***

```
ORDER BY Orders.orderdate DESC
```

Se la eseguiamo in ambiente Management Studio, otterremo la classica tabella di cui alla figura 1
Se, invece, aggiungiamo il costrutto FOR XML, così :

```
SELECT TOP 10
...
from Orders
inner join Customers
ON Orders.CustomerID=Customers.CustomerID
ORDER BY Orders.orderdate DESC
FOR XML AUTO
```

```
<Orders OrderID="11077" CustomerID="RATTC" OrderDa
  <Customers CompanyName="Rattlesnake Canyon Groce
</Orders>
<Orders OrderID="11076" CustomerID="BONAP" OrderDa
  <Customers CompanyName="Bon app'" Address="12, r
</Orders>
<Orders OrderID="11075" CustomerID="RICSU" OrderDa
  <Customers CompanyName="Richter Supermarkt" Addr
</Orders>
<Orders OrderID="11074" CustomerID="SIMOB" OrderDa
  <Customers CompanyName="Simons bistro" Address="
</Orders>
<Orders OrderID="11073" CustomerID="PERIC" OrderDa
  <Customers CompanyName="Pericles Comidas clásica
</Orders>
<Orders OrderID="11072" CustomerID="ERNSH" OrderDa
  <Customers CompanyName="Ernst Handel" Address="K
</Orders>
<Orders OrderID="11071" CustomerID="LILAS" OrderDa
  <Customers CompanyName="LILA-Supermercado" Addre
</Orders>
<Orders OrderID="11070" CustomerID="LEHMS" OrderDa
  <Customers CompanyName="Lehmanns Marktstand" Add
</Orders>
```

Fig. 2: ***esecuzione query XML***

Il risultato sarà un documento XML come mostrato in figura 2.
Accanto al costrutto FOR XML dobbiamo sempre impostare la modalità con cui SQL Server deve formattare l'output, le modalità sono AUTO, RAW ed EXPLICIT.

MODALITÀ DI OUTPUT

La modalità più semplice per definire l'output della query è AUTO. In questo caso SQL Server scrive la riga in un nodo dallo stesso nome della tabella, e i valori delle colonne come attributi, come in:

```
<Orders OrderID="11077" CustomerID="RATTC"
        OrderDate="1998-05-06T00:00:00"
        RequiredDate="1998-06-03T00:00:00"
        ShipName="Rattlesnake Canyon Grocery"
        ShipAddress="2817 Milton Dr."
        ShipCity="Albuquerque" ShipPostalCode="87110"
        ShipCountry="USA"> ...
```

È importante notare che, nel caso in cui nella query vi siano dei join ad altre tabelle, l'output risulterà nidificato, nel nostro caso :

```
<Orders OrderID="11077" ...>
  <Customers CompanyName="Rattlesnake Canyon
        Grocery" Address="2817 Milton Dr."
        City="Albuquerque" Country="USA"
        PostalCode="87110" />
</Orders>
```

In modalità AUTO non c'è modo di controllare questa nidificazione, mentre è possibile definire che i valori delle colonne vengano restituiti come elementi anziché come attributi. Tale modalità si imposta con l'opzione ELEMENTS, e cioè :

```
FOR XML AUTO,ELEMENTS
```

E dà luogo ad un output del tipo :

```
<Orders>
  <OrderID>11077</OrderID>
  <CustomerID>RATTC</CustomerID>
  <OrderDate>1998-05-06T00:00:00</OrderDate>
  <RequiredDate>1998-06
        03T00:00:00</RequiredDate>
  <ShipName>Rattlesnake Canyon
        Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
  <Customers>
    <CompanyName>Rattlesnake Canyon
        Grocery</CompanyName>
    <Address>2817 Milton Dr.</Address>
    <City>Albuquerque</City>
    <Country>USA</Country>
    <PostalCode>87110</PostalCode>
  </Customers>
</Orders>
```

La modalità RAW, invece produce un output con il nodo corrispondente alla riga chiamato sempre row e non nidifica le tabelle legate da join .

```
FOR XML RAW
```

Dà luogo a :

```
<row OrderID="11077" CustomerID="RATTC"
        OrderDate="1998-05-06T00:00:00"
        RequiredDate="1998-06-03T00:00:00"
        ShipName="Rattlesnake Canyon Grocery"
        ShipAddress="2817 Milton Dr."
```

SQL SERVER MANAGEMENT STUDIO

Di Sql Server esiste come sappiamo anche una versione gratuita chiamata Express (distribuita con il numero 103 di IoProgrammo). Non dimenticate però anche di installare anche lo strumento di gestione SQL Server Management Studio che rappresenta la sintesi dei tools Query Analizer e Enterprise Manager di SQL Server 2000. SQL Server Management Studio è anche scaricabile all'indirizzo: http://msdn.microsoft.com/vstudio/express/sql/download

```
ShipCity="Albuquerque" ShipPostalCode="87110"
ShipCountry="USA" CompanyName="Rattlesnake
Canyon Grocery" Address="2817 Milton Dr."
City="Albuquerque" Country="USA"
PostalCode="87110" />
```

Anche qui possiamo usare l'opzione ELEMENTS per avere un output con elementi:

```
<row>
  <OrderID>11077</OrderID>
  <CustomerID>RATTC</CustomerID>
  <OrderDate>1998-05-06T00:00:00</OrderDate>
  <RequiredDate>1998-06
                   03T00:00:00</RequiredDate>
  <ShipName>Rattlesnake Canyon
                   Grocery</ShipName>
  <ShipAddress>2817 Milton Dr.</ShipAddress>
  <ShipCity>Albuquerque</ShipCity>
  <ShipPostalCode>87110</ShipPostalCode>
  <ShipCountry>USA</ShipCountry>
  <CompanyName>Rattlesnake Canyon
                   Grocery</CompanyName>
  <Address>2817 Milton Dr.</Address>
  <City>Albuquerque</City>
  <Country>USA</Country>
  <PostalCode>87110</PostalCode>
</row>
```

```
<Customer CustomerID="ALFKI">
    <Order OrderID=10643>
   <Order OrderID=10692>
   ...
</Customer>
<Customer CustomerID="ANATR" >
    <Order OrderID=10308 >
   <Order OrderID=10625 >
   ...
</Customer>
```

Tale output è prodotto da una query del tipo:

```
SELECT 1  as Tag,
       NULL  as Parent,
       Customers.CustomerID as
                   [Customer!1!CustomerID],
       NULL  as [Order!2!OrderID]
FROM Customers

UNION ALL
SELECT 2,
       1,
       Customers.CustomerID,
       Orders.OrderID
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
ORDER BY [Customer!1!CustomerID],
                   [Order!2!OrderID]
```

Tag	Parent	Customer!1!CustomerID	Order!2!OrderID
1	NULL	ALFKI	NULL
2	1	ALFKI	10643
2	1	ALFKI	10692
2	1	ALFKI	10702
2	1	ALFKI	11011
2	1	ALFKI	...

Tabella 1: La tabella restituita con l'opzione explicit

SUPPORTO XML IN SQL SERVER 2005

Una buona risorsa per approfondire le novità introdotte nel supporto nativo a XML da SQL Server 2005 è l'articolo pubblicato su http://www.dotnethell.it/articles/SQL-Server-2005-XML.aspx .

Le modalità AUTO e RAW non impattano sulla query di base, mentre per utilizzare la modalità EXPLICIT è necessario formattare la query in modo da creare una tabella c.d. universale.

La tabella universale deve avere i seguenti requisiti:

- La prima colonna specificata nella clausola SELECT deve essere un numero di tag (Tag) definito.
- La seconda colonna specificata deve essere un numero di tag (Parent) definito dell'elemento padre.

Prendiamo un esempio (tratto dalla documentazione Microsoft). L'output che dobbiamo ottenere è :

```
FOR XML EXPLICIT
```

Che corrisponde alla tabella :
La modalità EXPLICIT è senz'altro quella più complessa da comprendere e da ottenere, tuttavia consente la massima precisione in fatto di formattazione finale, per cui vale la pena di approfondirla.

SQL SERVER 2005

I costrutti che abbiamo visto sono utilizzabili già a partire da SQL Server 2000, tuttavia la versione 2005 presenta molte, interessanti novità. In primo luogo c'è da dire che l'output XML prodotto dalle query che abbiamo visto non produce XML sintatticamente valido, ma piuttosto un frammento di XML. Ciò avviene perché man-

ca un elemento radice univoco in quanto al primo livello troviamo sempre un nodo per ogni riga della tabella, ciò ci costringeva a rielaborare, in sede di programma l'output ottenuto per inserirlo in un elemento di primo livello. SQL Server 2005 ci permette invece di dichiarare l'elemento di primo livello in cui inserire l'output:

```
SELECT TOP 10
...
from Orders
inner join Customers
ON Orders.CustomerID=Customers.CustomerID
ORDER BY Orders.orderdate DESC
FOR XML AUTO, ROOT('document')
```

Produrrà quindi:

```
<document>
  <Orders OrderID="11077" CustomerID="RATTC"
          OrderDate="1998-05-06T00:00:00"
          RequiredDate="1998-06-03T00:00:00"
          ShipName="Rattlesnake Canyon Grocery"
          ShipAddress="2817 Milton Dr."
          ShipCity="Albuquerque" ShipPostalCode="87110"
          ShipCountry="USA">
    <Customers CompanyName="Rattlesnake Canyon
          Grocery" Address="2817 Milton Dr."
          City="Albuquerque" Country="USA"
          PostalCode="87110" />
  </Orders>
...
</document>
```

IL TIPO DI OUTPUT

C'è poi da dire che l'output in SQL Server 2000, era necessariamente una stringa di lunghezza massima di 4000 caratteri, se il documento risultante fosse stato più lungo la query avrebbe suddiviso il testo tra in più righe che noi, in fase di lettura con un DataReader, avremmo dovuto ricomporre in un'unica stringa prima di interpretarne il risultato, come avviene in questa funzione che restituisce una stringa, dove la variabile reader rappresenta un oggetto SqlDataReader :

```
    Public Function ReadForXml(ByVal reader As
SqlDataReader, Optional ByVal RootName As String =
                      "DocumentElement") As String
        Dim sb As New System.Text.StringBuilder
        sb.AppendFormat("<{0}>", RootName)
        While reader.Read
            sb.Append(reader.GetSqlString(0))
        End While
        reader.Close()
        sb.AppendFormat("</{0}>", RootName)
        Return sb.ToString
    End Function
```

Sql Server 2005, invece, ha XML come proprio dataType, quindi è possibile estrarre i dati direttamente come type XML specificando l'opzione TYPE nella clausola FOR XML come in:

```
SELECT TOP 10
...
from Orders
inner join Customers
ON Orders.CustomerID=Customers.CustomerID
ORDER BY Orders.orderdate DESC
FOR XML AUTO, TYPE, ROOT('document')
```

In questo caso basta richiamare il metodo ExecuteXmlReader dell'oggetto SqlCommand per ottenere direttamente un XmlReader:

```
Dim reader As XmlReader = cmd.ExecuteXmlReader
```

Le innovazioni apportate alle query in XML da Sql Server 2005 sono molte altre, dalla possibilità di nidificare le clausole FOR XML a quella di specificare uno schema XSD, ma passiamo ora direttamente a risolvere il nostro problema iniziale: tabella ordini e dettaglio fattura.

MODALITÀ EXPLICIT

Un'illustrazione completa delle query SQL con la clausola FOR XML EXPLICIT richiederebbe un intero articolo. Al tempo stesso, dopo aver familiarizzato con il costrutto FOR XML in modalità AUTO o RAW, sentirete ben presto l'esigenza di un maggior controllo sull'output XML che riuscirete ad ottenere solo con EXPLICIT.
Per fortuna l'opzione è ben documentata nella documentazione di Sql Book Online che viene distribuita da Microsoft insieme a Sql Server, in tutte le versioni.

LA NOSTRA APPLICAZIONE

In Visual Studio 2005 (anche Express) creiamo un nuovo progetto Web dove inseriremo due nuove pagine ASP.NET default.aspx (tabella ordini) e fattura.aspx (dettaglio fattura).
Creiamo anche un nuovo file di codice, che chiameremo common.vb.
Per questo esempio, infatti, sarà sufficiente scrivere il nostro codice condiviso in common.vb che poi può essere incluso nelle pagine ASP.NET attraverso il tag script in questo

modo:

```
<script runat="server" src="common.vb"></script>
```

Questo ci eviterà di dover scrivere lo stesso codice per ogni pagina. Da ricordare che il codice che inseriamo in common.vb è già nel contesto della pagina e quindi non dovrà essere inserito in una dichiarazione di classe.
Il meccanismo di funzionamento del programma sarà il seguente:

1. lettura del testo della query da un file su disco
2. esecuzione della query per ottenere un documento XML
3. trasformazione del documento XML con un foglio di stile XSL anch'esso salvato su disco per ottenere una stringa
4. assegnazione della stringa risultante ad un controllo DIV marcato come runat="server"

Nel corpo di common.vb inseriamo una costante che rappresenta la stringa di connessione:

```
Private Const CONNECTIONSTRING As String = "Data
                Source=localhost\SQLEXPRESS;Initial
        Catalog=Northwind;Integrated Security=True"
```

Naturalmente in ambiente di produzione sarebbe preferibile inserire il valore nel file di configurazione.
La funzione che legge il file contenente la query su disco e ne ricava un documento XML sarà la seguente:

```
Private Function ReadXPathDocument(ByVal
 queryFilename As String, ByVal ParamArray args() As
                         String) As XPathDocument
   Dim conn As New
              SqlConnection(CONNECTIONSTRING)
   Dim cmd As SqlCommand =
                     conn.CreateCommand()
   Dim sqlText As String =
               FileIO.FileSystem.ReadAllText
                     (MapPath(queryFilename))
   cmd.CommandText = String.Format(sqlText, args)
   cmd.CommandType = CommandType.Text
   conn.Open()
   Dim reader As XmlReader =
                        cmd.ExecuteXmlReader
   Dim doc As New XPathDocument(reader)
   conn.Close()
   Return doc
End Function
```

È da notare il parametro *args*() della funzione che ci consente di inserire nel corpo della query SQL dei segnaposto tipo {0} che saranno sostituiti, da *String.Format*, dai valori passati come argomenti.
Sempre in common.vb scriviamo la funzione che si occupa di fare tutto il lavoro di trasformazione :

```
Private Function GetOutput(ByVal queryFilename As
             String, ByVal xsltFilename As String, ByVal
                ParamArray args() As String) As String
   Dim xslEngine As New XslCompiledTransform()
   xslEngine.Load(MapPath(xsltFilename),
              XsltSettings.Default, New XmlUrlResolver)
   Dim sb As New System.Text.StringBuilder
   Dim writer As New IO.StringWriter(sb)
xslEngine.Transform(ReadXPathDocument(queryFilen
                      me, args), Nothing, writer)
   Return sb.ToString
End Function
```

In pratica, carica in un oggetto XslCompiled Transform il file XSLT da disco, recupera i dati con la funzione *ReadXPathDocument* che abbiamo visto prima e, infine, restituisce come stringa il risultato della trasformazione.
A questo punto scriviamo il testo delle query in due files : Ordini.sql e Fattura.sql (trovate il testo, piuttosto lungo, nell'esempio allegato al cd).
Il codice delle pagine ASP.NET è davvero minimo, vediamo quello di default.aspx:

```
<%@ Page Language="VB"  %>
<script runat="server" src="common.vb"></script>
<script runat="server">
   Protected Sub Page_Load(ByVal sender As Object,
                     ByVal e As System.EventArgs)
      tablePlaceHolder.InnerHtml =
                GetOutput("Ordini.sql", "Ordini.xsl")
   End Sub
</script>
<html>
<head><title>Ordini</title></head>
<body>
   <div id="tablePlaceHolder"
                      runat="Server"></div>
</body>
```

ASP.NET IN VISUAL STUDIO

Visual Studio 2005 offre, a differenza della versione 2003, il pieno supporto allo sviluppo di pagine con codice inserito direttamente nel tag <script>, offrendo anche in questo caso tutte le funzionalità di intellisense. Questa è stata una novità salutata con favore dagli sviluppatori provenienti da ASP classico abituati a lavorare con il codice "inline".

```
</html>
```

In pratica si richiama, nel gestore dell'evento Load della pagina, la funzione GetOutput passando il nome del file con la query SQL e quello del file XSL.
Tutta la logica di trasformazione, a questo punto, si sposta sul file XSL che trasforma l'input XML ricevuto da SQL Server.
Vediamo, ad esempio, il punto saliente del file Ordini.xsl lì dove si crea la riga della tabella Ordini:

```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
>
...
<xsl:template match="row">
        <tr class="row"
onmouseover="this.className='rowSel'"
onmouseout="this.className='row'"
onclick="window.open('fattura.aspx?OrderId={@OrderID}')">
                <td><xsl:value-of
                        select="@OrderID"/></td>
                <td><xsl:value-of
                        select="@CompanyName"/></td>
                <td>
                        <xsl:value-of
                        select="@Address"/> <xsl:value-of
        select="@ShipPostalCode"/> - <xsl:value-of
                        select="@City"/>
                        (<xsl:value-of
                        select="@Country"/>)
                </td>
                <td align="right"><xsl:value-of
                        select="@OrderDate"/></td>
        </tr>
</xsl:template>
...
</xsl:stylesheet>
```

Notiamo che abbiamo fatto in modo, utilizzando javascript, che la fattura si apra con un clic sull'intera riga e che è stato implementato un effetto mouseover sulla riga, cose che ... provatevi a fare con una DataGrid!

IL RISULTATO

Una volta preparate le query e i fogli di stile, non resta che azionare il famoso F5 dall'ambiente di sviluppo per far partire il web server integrato e il browser alla pagina default.aspx. Vediamo quindi il riepilogo ordini (figura 3) con le righe cliccabili che azionano un popup con il dettaglio della fattura (figura 4)

Ordine	Cliente	Indirizzo cliente	Data ordine
11077	Rattlesnake Canyon Grocery	2817 Milton Dr.87110 - Albuquerque (USA)	1998-05-06T00:00:00
11076	Bon app'	12, rue des Bouchers13008 - Marseille (France)	1998-05-06T00:00:00
11075	Richter Supermarkt	Grenzacherweg 2371204 - Genève (Switzerland)	1998-05-06T00:00:00
11074	Simons bistro	Vinbæltet 341734 - Kobenhavn (Denmark)	1998-05-06T00:00:00
11073	Pericles Comidas clásicas	Calle Dr. Jorge Cash 32105033 - México D.F. (Mexico)	1998-05-05T00:00:00
11072	Ernst Handel	Kirchgasse 68010 - Graz (Austria)	1998-05-05T00:00:00
11071	LILA-Supermercado	Carrera 52 con Ave. Bolívar #65-98 Llano Largo3508 - Barquisimeto (Venezuela)	1998-05-05T00:00:00
11070	Lehmanns Marktstand	Magazinweg 760528 - Frankfurt a.M. (Germany)	1998-05-05T00:00:00
11069	Tortuga Restaurante	Avda. Azteca 12305033 - México D.F. (Mexico)	1998-05-04T00:00:00
11068	Queen Cozinha	Alameda dos Canàrios, 89105487-020 - Sao Paulo (Brazil)	1998-05-04T00:00:00

Fig. 3: **Riepilogo ordini**

FATTURA: 11076

Spett.le
Bon app'
12, rue des Bouchers
13008 - Marseille
France

Prodotto	Prezzo	Quantità	Totale
Grandma's Boysenberry Spread	25,00	20	500,00
Tofu	23,25	20	465,00
Teatime Chocolate Biscuits	9,20	10	92,00
	57,45	50	2.872,50

Fig. 3: **Dettaglio fattura**

CONSIDERAZIONI FINALI

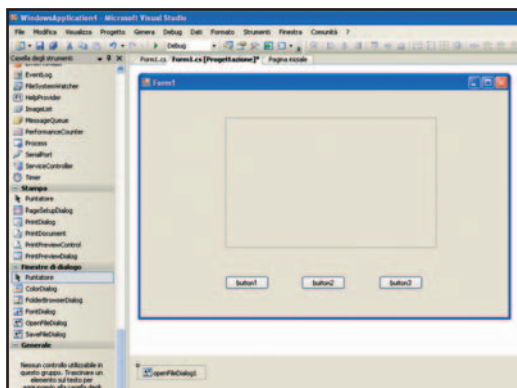
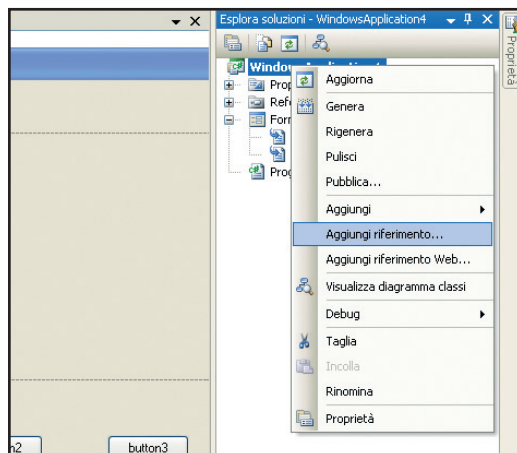
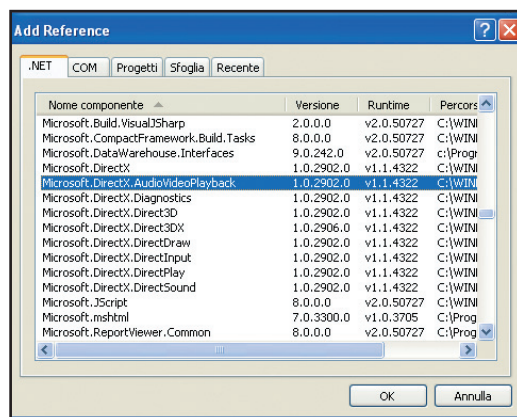
XMl è uno strumento che ha già qualche anno di vita e molte delle applicazioni moderne ne fanno uso. Non a caso SQL Server 2005 integra nativamente le estensioni per XML. Il poter ottenere i risultati di una Query direttamente in questo formato, fa si che molto dell'eventuale lavoro di trasformazione sia demandato direttamente al server, con la logica conseguenza di una netta diminuizione del numero di righe di codice, e di eventuali errori.ß
Da prove effettuate, la trasformazione diretta SQL/XSL si è rivelata dal 10% al 50% più veloce rispetto all'utilizzo del controllo DataGrid, un buon motivo per approfondire questa tecnica che, inoltre, lascia piena libertà allo sviluppatore nel controllo dell'output prodotto.

Francesco Smelzo

COME POSSO VISUALIZZARE UN FILMATO IN .NET

LA COSA PIÙ SEMPLICE È PASSARE ATTRAVERSO LE DIRECTX. VEDIAMO COME.

FACCIAMOLO IN C#

1 Su una form posizioniamo tre bottoni, un pannello e un componente OpenFileDialog

2 Usiamo il menu "aggiungi riferimento" per poter utilizzare le DirectX

3 Dalla finestra che compare selezioniamo "Microsoft AudioVideoPlayback"

4 Aggiungiamo un oggetto privato di classe video

```
private Video _video;
```

5 clicchiamo due volte nel buttone uno per implementare il codice relativo all'evento onclick

```
if (openFileDialog1.ShowDialog() ==
                    DialogResult.OK)
    {
        int height = panel1.Height;
        int width = panel1.Width;
        _video = new
            Video(openFileDialog1.FileName);
        _video.Owner =panel1;
        panel1.Width = width;
        panel1.Height = height;
        _video.Play();
    }
```

6 Programmiamo anche il pulsante due che deve mettere in pausa il video

```
private void button2_Click
            (object sender, EventArgs e)
    {
        _video.Pause();
    }
```

7 Possiamo anche programmare il pulsante tre per far si che in seguito alla sua pressione il video venga riprodotto

```
private void button3_Click
            (object sender, EventArgs e)
    {
        _video.Play();
    }
```

COME FUNZIONA?

A fare quasi tutto il lavoro sono le DirectX. Nella parte iniziale aggiungiamo un riferimento per poterle utilizzare all'interno del nostro progetto. Il codice è poi abbastanza semplice, è sufficiente creare un container in cui proiettare il video per poi programmare i vari pulsanti tramite i metodi appropriati. Creiamo un nuovo oggetto di classe _video, gli assegnamo come owner un panel. Utilizziamo il metodo play per proiettare il video

nel pannello, altri metodi possibili sono ad esempio lo stop o il rewind.

FACCIAMOLO CON VISUAL BASIC

1 i passi da uno a tre rimangono sostanzialmente identici. La dichiarazione dell'oggetto video invece diventa

```
Dim _video As Video
```

2 Il bottone play diventa

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                              Handles Button1.Click
    If (OpenFileDialog1.ShowDialog() =
                              DialogResult.OK) Then
        Dim height As Integer
        Dim width As Integer
        _video = New
                     Video(OpenFileDialog1.FileName)
        width = Panel1.Width
        height = Panel1.Height
        _video.Owner = Panel1
        Panel1.Width = width
```

```
        Panel1.Height = height
        _video.Play()
    End If
End Sub
```

3 Il bottone play diventa

```
Private Sub Button3_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                              Handles Button3.Click
    _video.Play()
End Sub
```

4 Il bottone pause diventa

```
Private Sub Button3_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                              Handles Button3.Click
    _video.Pause()
End Sub
```

DOVE POSSO TROVARE LE DIRECTX ?

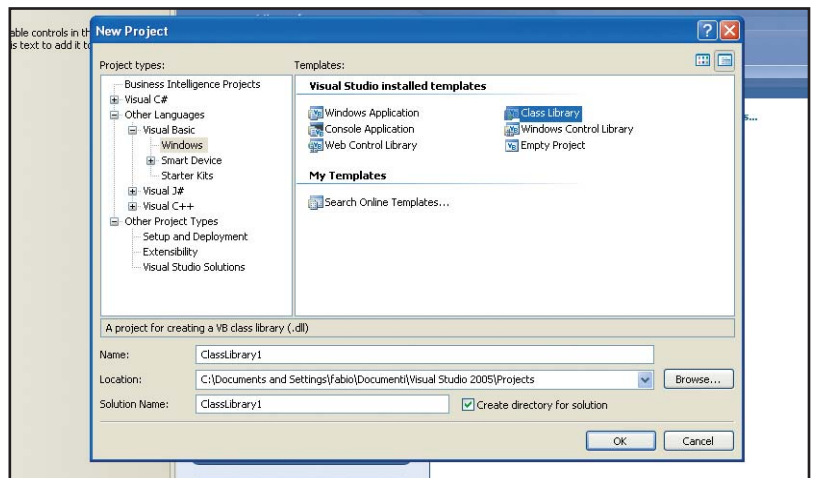
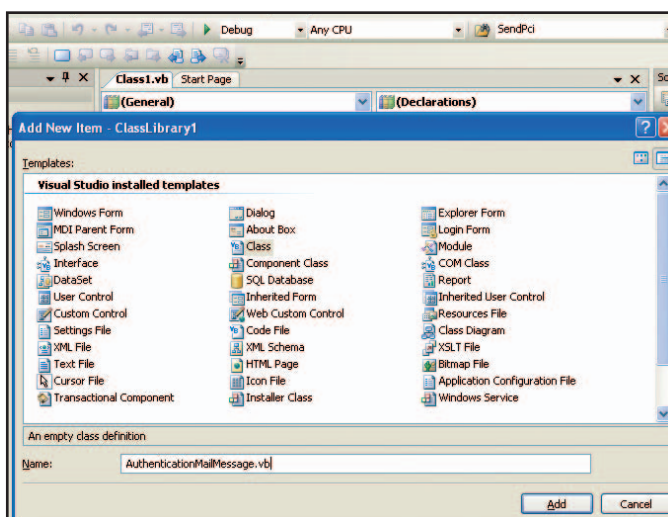
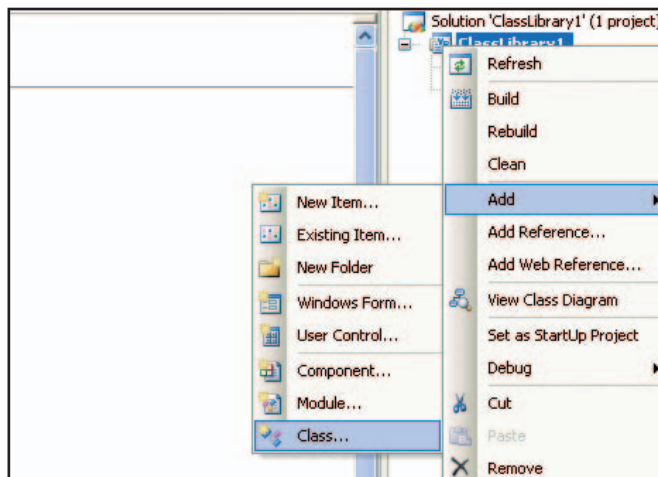
Perché tutto funzioni avrete dovuto installare le DirectX almeno nella versione 9. Potete scaricarle all'indirizzo http://msdn.microsoft.com/directx/sdk

COME POSSO CREARE ISTANZE A RUNTIME?

IMPLEMENTIAMO UNA CLASSE BASE. CREEREMO TUTTE LE ALTRE DERIVANDOLA DA QUESTA E INSTANZIANDOLA CON UN METODO EXECUTE GENERICO

Talvolta abbiamo la necessità di utilizzare delle classi di cui non conosciamo la posizione in fase di sviluppo. In questo caso dobbiamo implementare un sistema che ci garantisca una tipizzazione, ma anche la possibilità di creare istanze a runtime. Vediamo come implementare un piccolo sistema di plug-in utilizzando Visual Basic 2005. Il nostro sistema utilizzerà una classe base IoBase che tutte le classi non conosciute dovranno implementare. IoBase avrà il solo metodo CreateInstance.

FACCIAMOLO CON VISUAL BASIC

1 Creiamo un nuovo progetto Console Application utilizzando il linguaggio Visual Basic .NET

2 Aggiungiamo una classe, la chiamiamo IoBase.vb

3 e la dichiariamo come MustInherit (abstract in C#).

```
Public MustInherit Class IoBase
End Class
```

4 Inseriamo anche l'unico metodo della classe, il metodo Execute

```
Function execute()
    End Function
```

5 Inseriamo il metodo CreateInstance e lo dichiariamo shared (static in C#):

```
    Public Shared Function CreateInstance() As
                                          IoBase
        Dim asm As Assembly
        Dim strType As String
        Dim arrayType() As String
        strType =
  ConfigurationManager.AppSettings("CurrentClass")
        If (strType.IndexOf(",") > 0) Then
            arrayType = strType.Split(",")
            strType = arrayType(0)
            asm =
                Assembly.Load(arrayType(1).Trim())
        Else
            asm =
                  Assembly.GetExecutingAssembly()
        End If

        Return
    CType(asm.CreateInstance(strType),IoBase)
```

6 il metodo CreateInstance richiede la presenza nel file di configurazione di una chiave CurrentClass che indica qual è il tipo da istanziare ed eventualmente in che assembly si trova:

```
<configuration>
  <appSettings>
    <add key="CurrentClass"
value="ioProgrammo.FirstClass, ioProgrammo"/>
  </appSettings>
</configuration>
```

Il tipo ioProgrammo.FirstClass si trova nell'assembly ioProgrammo. Il codice sopraindicato legge il valore e verifica la presenza del nome dell'assembly. Se trovato esegue lo split della stringa e crea la relativa istanza, altrimenti recupera l'istanza dell'assembly corrente

7 l'implementazione del client sarà simile a questa:

```
    Dim io As IoBase =
                     IoBase.CreateInstance()
    Dim s As String = io.Execute()
```

Semplicemente modificando il tipo e l'assembly indicati nel .config possiamo sostituire la classe inizialmente dichiarata con un'altra che implementa sempre la classe base IoBase.

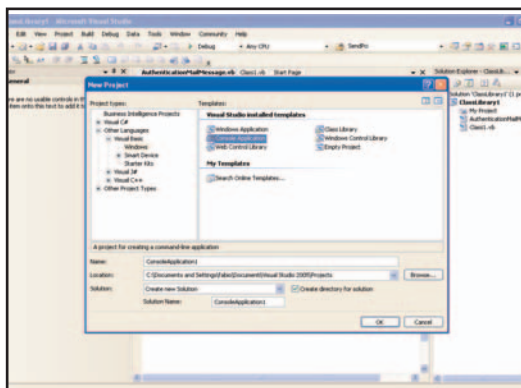
COME FUNZIONA?

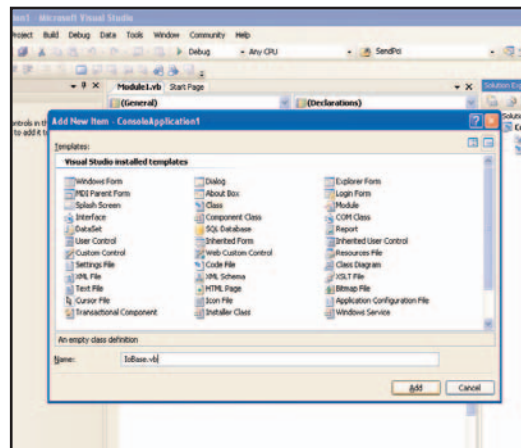
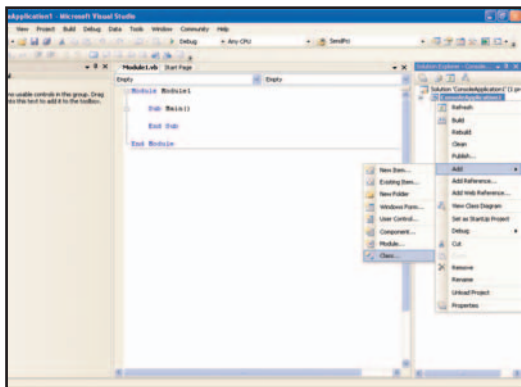
È abbastanza semplice. Le informazioni relative alla classe vengono tratte dal file XML. Il file viene parserizzato alla ricerca del parametro CurrentrClass, se lo trova splitta la stringa e crea l'istanza. Il metodo è abbstanza comodo quando si devono creare classi rapidamente senza manipolare il codice, la funzione è quella di creare una factory abbastanza dinamica senza troppi problemi di programmazione modificando esclusivamente il file contenente la definizione delle classi

COME POSSO INVIARE MESSAGGI E-MAIL CON AUTENTICAZIONE?

NELLA VERSIONE 1.1 DEL FRAMEWORK È RELATIVAMENTE SEMPLICE PREPARARE ED INVIARE MESSAGGI EMAIL MA È UN PO MENO FACILE GESTIRE LE MODALITÀ DI AUTENTICAZIONE SUL SERVER SMTP. VEDIAMO COME È POSSIBILE FARLO UTILIZZANDO VISUAL BASIC .NET.

FACCIAMOLO CON VISUAL BASIC

1 Creiamo un nuovo progetto Visual Basic Class Library

2 Aggiungiamo la classe AuthenticationMailMessage.vb

VISUAL BASIC.NET

3 Inseriamo il seguente codice:

```
Public Class AuthenticationMailMessage
            Inherits MailMessage
            Public Property WriteOnly Credentials As
System.Net.NetworkCredentials
                        Set
MyBase.Fields("http://schemas.microsoft.com/cdo/
                onfiguration/smtpauthenticate") = 1;
MyBase.Fields("http://schemas.microsoft.com/cdo/
      onfiguration/sendusername") = value.UserName;
MyBase.Fields("http://schemas.microsoft.com/cdo/
      onfiguration/sendpassword") = value.Password;
                        End Set
            End Public
End Public
```

il primo parametro imposta il tipo di autenticazione. I possibili valori sono: 0 – nessuna, 1 – Basic, 2 – NTLM.

4 Utilizziamo ora la classe per l'invio di un messaggio e-mail:

```
` Imposto il messaggio e-mail da inviare

Dim mail As New AuthenticationMailMessage()
mail.From = "fabio@dotnetside.org"
mail.To = "fabio@dotnetside.org"
mail.Subject = "Prova di un messaggio e-mail"
mail.Body = "Questo è un messaggio e-mail di
                                          prova."
` Imposto le credenziali per l'invio
mail.Credentials = New
           System.Net.NetworkCredentials("username",
                                      "password")
` Imposto il server smtp e invio il messaggio

SmtpMail.SmtpServer = "mail.server.com"
SmtpMail.Send(mail)
```

In questo modo garantiamo l'autenticazione per l'invio di e-mail anche per quei server che lo richiedono. Il funzionamento è c hiaro, tutto è demandato al metodo Credentials, che riceve username e password e provvede ad inviare le aL server le informazioni con le credenziali corrette per l'autenticazione

SQL SERVER 2005 SERVICE BROKER

SI TRATTA DI UNA DELLE FUNZIONALITÀ PIÙ INNOVATIVE NELLA NUOVA VERSIONE DEL DB DI MICROSOFT. IMPLEMENTA UN SISTEMA DI MESSAGISTICA BASATA SU CODE, FRA SERVER. VEDREMO COME SFRUTTARE QUESTA CARATTERISTICA PER LE NOSTRE APPLICAZIONI

Il nostro problema di oggi è semplice. Abbiamo mandato in giro per l'Italia i nostri agenti di vendita. Li abbiamo equipaggiati con un notebook e una fantastica applicazione che consente di registrare gli ordini in un database. Ora, va da sè che se ciascun notebook utilizza un suo database, la nostra azienda non avrà mai una visione globale degli ordini registrati dagli utenti. Abbiamo bisogno di un database centrale, in cui riversare i dati prelevati dai singoli database degli agenti. Il problema non è semplice, perché i nostri agenti possono star fuori anche qualche giorno prima di tornare in sede e ovviamente non dispongono perennemente di una connessione internet, per cui non possono inviare i dati in tempo reale al server centrale. Devono necessariamente immagazzinarli in un DB interno per poi scaricarli sul DB centrale alla prima occasione utile. Come fare? Ci viene in aiuto "Service Broker". Si tratta di un servizio di "messaggistica" asincrona implementato in Microsoft SQL server 2005. In sostanza il nostro problema sarà risolto in questo modo. Sul notebook del cliente installeremo SQL Server express edition, che come tutti sanno è gratuito ed è più che sufficiente per le esigenze del singolo agente. Il nostro server remoto sarà invece un SQL Server 2005 standard edition. Quando l'agente salverà i dati, in realtà li "ingloberà" in un messaggio XML che sarà conservato in una coda nel database interno. Alla prima disponibilità di connessione, il messaggio contenente il dato da salvare sarà inviato al server centrale che lo riceverà e lo elaborerà inserendo il dato. Tutte le modalità per lo scambio di messaggi sono implementate in un servizio denominato "Service Broker". Ovviamente è possibile scambiare messaggi di qualunque tipo, quindi al di là del semplice esempio che abbiamo proposto per rendere immediata la comprensione dell'argomento, vedremo che un servizio di messaggistica interno a SQL server può essere utilizzato in più di una occasione.

SERVICE BROKER IN PROFONDITÀ

È una delle novità più interessanti introdotte da SQL Server 2005. Come già detto si tratta di un sistema per lo scambio asincrono di messaggi tra servizi. Quando un'applicazione invia un messaggio ad un servizio, Service Broker lo inserisce in una coda associata al servizio di destinazione che lo leggerà quando avrà risorse. I messaggi possono essere inviati e ricevuti dallo stesso database, da database diversi o anche da database remoti.
Service Broker consente a due endpoint di comunicare grazie ad un protocollo chiamato Dialog che gestisce la comunicazione asincrona e garantisce il corretto ordine di ricezione dei messaggi utilizzando delle estensioni di T-SQL per la gestione delle code.
Service Broker ha un'architettura a livelli. In cima alla pila, a livello applicativo abbiamo due entità che vogliono scambiarsi messaggi.

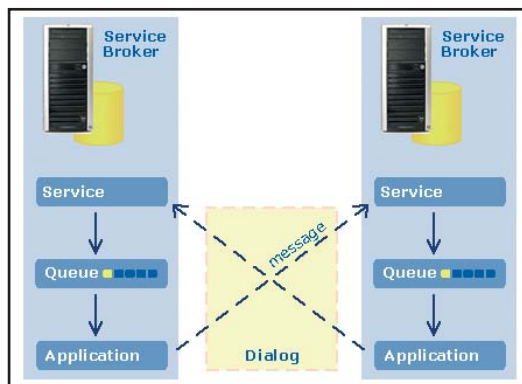



Fig. 1: Architettura di Service Broker

Pensate ad esempio ad un negozio online che riceve molti ordini. Piuttosto che salvare tutti i dati immediatamente nel server, gli ordini potrebbero essere immagazzinati un una coda ed elaborati sequenzialmente. In questo caso i nostri endpoint sarebbero il negozio on-line e lo storage server.
Abbiamo poi un livello intermedio, detto dei meta-dati, che serve a descrivere le modalità con cui le applicazioni parlano (i tipi di mes-

saggi che possono scambiarsi ed in che modalità possono farlo).
Il livello più basso, quello fisico, si occupa del trasporto e dell'accodamento dei messaggi.
Vediamo in dettaglio i componenti principali dell'architettura di Service Broker.

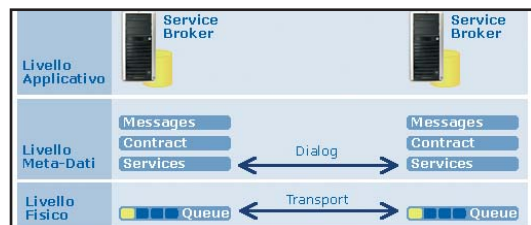



Fig. 2: *I livelli di trasporto*

COME SONO STRUTTURATI I MESSAGGI?

I messages sono le entità che vengono inviate con Service Broker. Ogni messaggio fa parte di una conversation. Possiamo inviare diversi tipi di messaggio come ad esempio XML o in formato binario. Quello che è possibile inviare viene definito tramite la creazione di Message Types.

Per creare un message type utilizziamo la seguente sintassi:

```
CREATE MESSAGE TYPE nome_del_message_type
    [ AUTHORIZATION nome_proprietario ]
    [ VALIDATION =
{ NONE
                | EMPTY
                | WELL_FORMED_XML
                | VALID_XML WITH SCHEMA COLLECTION
 Il_nome_dello_schema_collection
              } ]
```

Ciascun servizio "espone" un contract che definisce quali tipi di messaggi sono consentiti, chi può inviarli e chi può riceverli. La sintassi:

```
CREATE CONTRACT nome_contratto
   [ AUTHORIZATION nome_proprietario ]
      (  {   { nome_message_type   | [ DEFAULT ] }
          SENT BY { INITIATOR | TARGET | ANY }
      } [ ,...n] )
```

COME SONO STRUTTURATE LE CODE?

Le queues servono ad immagazzinare i messaggi in arrivo, sono associate ad un service program che rappresenta una stored procedure da eseguire quando il messaggio viene letto e processato.
Per definire una coda

```
CREATE QUEUE
              nome_database.nome_schema.nome_coda
   [ WITH
     [ STATUS = { ON | OFF }  [ , ] ]
     [ RETENTION = { ON | OFF } [ , ] ]
     [ ACTIVATION (
         [ STATUS = { ON | OFF } , ]
           PROCEDURE_NAME =
                         la_stored_prcedure_da_usare ,
           MAX_QUEUE_READERS =
                          numer_massimo_di_readers ,
           EXECUTE AS { SELF | 'nome_utente' |
                                            OWNER }
            ) ]
   ]
     [ ON { nome_filegroup | [ DEFAULT ] } ]
```

CHI INVIA E RICEVE I MESSAGGI?

I service sono utilizzati da Service Broker per inoltrare i messaggi alla relativa coda nel database. Un service program è l'entità che processa i messaggi; viene attivato quando arriva un messaggio dalla coda. A ciascun service è associato un contract, in modo che i messaggi possano essere validati.

La sintassi è la seguente:

```
CREATE SERVICE nome_service
   [ AUTHORIZATION nome_proprietario ]
   ON QUEUE [ nome_schema. ]nome_coda
   [ ( nome_contract | [DEFAULT] [ ,...n ] ) ]
```

MESSAGGISTICA DISTRIBUITA

Le routes, sono utilizzate da Service Broker per capire dove inoltrare i messaggi nel caso in cui si utilizzino dei servizi di messaggistica distribuiti.
Per creare una rotta occorre specificare: il servizio che utilizzerà la rotta, l'indirizzo ed il protocollo da utilizzare. In T-SQL

```
CREATE ROUTE nome_rotta
[ AUTHORIZATION nome_ proprietario]
WITH
  [ SERVICE_NAME = 'nome_service', ]
  [ BROKER_INSTANCE = 'identificatore_istanza' , ]
  [ LIFETIME = tempo_di_vita , ]
  ADDRESS =  'indirizzo'
```

SECURITY

Service Broker fornisce meccanismi di sicurezza a livello di dialogo e di trasporto. A livello di dialogo, cripta i messaggi e verifica le identità dei partecipanti. A livello di trasporto, stabilisce una connessione autenticata tra due entità in modo da prevenire accessi non autorizzati.

```
[ , MIRROR_ADDRESS = 'indirizzo_mirror' ]
```

Un dialog è una conversazione tra due endpoint: l'initiator, il target.vPerché la conversazione avvenga è necessario specificare il contract che verrà utilizzato per conversare. Una conversation group è un insieme di dialogs. Il transport è il protocollo sottostante che permette a due endpoint di comunicare. E' un protocollo proprietario nativo di SQL Server 2005.

METTIAMO TUTTO INSIEME

Per utilizzare Service Broker occorre creare una serie di componenti necessari per il funzionamento del sistema. Innanzitutto, occorre creare dei message type, ovvero i tipi di messaggio che due endpoint potranno scambiarsi. E' necessario poi definire come avverrà la conversazione tramite un contract. Quindi, devono essere realizzate le queue per contenere i messaggi inviati durante la conversazione. Creati gli oggetti base dell'architettura, possiamo definire i services che permetteranno di scrivere e leggere dalle code. A questo punto, siamo in grado di iniziare un dialog tra due entità di Service Broker che prendono il nome di Initiator e Target.

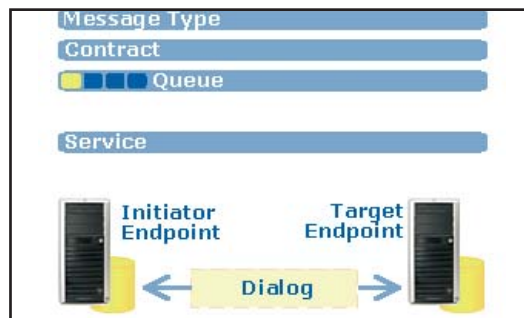



Fig. 3: **Le fasi di creazione di una conversazione**

IL CLASSICO "HELLO WORLD"

Mettiamo subito in pratica quello che abbiamo imparato con un semplice esempio di invio/ricezione messaggi. Possiamo eseguire le query a blocchi (fino ad ogni istruzione "go" per vedere cosa succede).

```
--creiamo il database di test
CREATE DATABASE TestServiceBroker
go
USE TestServiceBroker
go
--trustworthy deve essere impostata ad on per usare Service Broker
ALTER DATABASE TestServiceBroker SET trustworthy ON
go
--Occorre creare una master key
CREATE MASTER KEY ENCRYPTION BY PASSWORD =
'MorpheusWeb2006'
go
-- Il message type
CREATE MESSAGE TYPE myMessage
go
-- Il contract
CREATE CONTRACT myContract(myMessage SENT BY ANY)
go
-- Creiamo le code per invio e ricezione dei messaggi
CREATE QUEUE SenderQueue
CREATE QUEUE ReceiverQueue
go
-- I services per inviare e ricevere
CREATE SERVICE SenderService ON QUEUE SendeQueue
CREATE SERVICE ReceiverService ON QUEUE ReceiverQueue (myContract)
go
-- Lanciamo la query in un'altro query browser,
--resterà in attesa dell'arrivo di messaggi in coda
WAITFOR (RECEIVE TOP(1)
  CAST(message_body as XML)
   FROM ReceiverQueue);
go

-- Torniamo alla precedente finestra ed inviamo dei messaggi
-- creiamo un dialog con un id di conversazione ed inviamo dal servizio sender al receiver.
-- Quando abbiamo inviato tutti i messaggi terminiamo la conversazione
DECLARE @id uniqueidentifier
BEGIN DIALOG CONVERSATION @id
 FROM SERVICE SenderService TO SERVICE 'Receive Service'
 ON CONTRACT myContract;
```

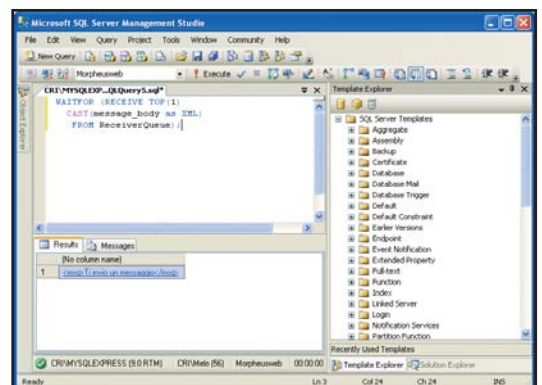



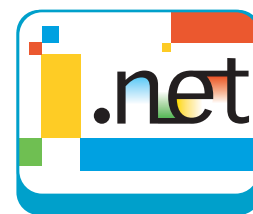
Fig. 4: **La ricezione del messaggio**

```
SEND ON CONVERSATION @id MESSAGE TYPE m
        Message ('<msg>Ciao</msg>');
SEND ON CONVERSATION @id MESSAGE TYPE
        myMessage ('<msg>Carmelo</msg>');
SEND ON CONVERSATION @id MESSAGE TYPE m
        Message ('<msg>Come va?</msg>');
END CONVERSATION @id WITH CLEANUP;
Go
```

Fig. 6: *I templates di SQL Server Magement Studio*

SQL SERVER MANAGEMENT STUDIO

Nel caso di utilizzo di SQL Server Management Studio, abbiamo la possibilità di visualizzare i componenti di Service Broker per ciascuno dei nostri database.
Sql Server Express non ha una gestione "visua-

Fig. 5: *SQL Server Management Studio*

le" di Service Broker: possiamo comunque visualizzare gli oggetti che ci interessano tramite delle view di sistema come:

```
sys.service_message_types
sys.service_contracts
sys.service_contract_message_usages
sys.service_contract_usages
sys.service_queues
sys.service_queue_usages
sys.services
sys.conversation_groups
sys.conversation_endpoints
```

Per visualizzare i message types, ad esempio, scriveremo

```
SELECT  * FROM sys.service_message_types
```

SQL Server Management Studio contiene anche molti template per la creazione di oggetti di Service Broker. Basta cliccare su "*View*"->"*Template* Explorer" per visualizzare la lista dei template disponibili per numerose funzionalità di SQL Server tra cui Service Broker.

QUANDO USARE SERVICE BROKER

Molte applicazioni che fanno uso dei trigger potrebbero utilizzare Service Broker per realizzare dei trigger asincroni. Il trigger potrebbe inserire un messaggio in coda invece di processarlo subito in modo da poter continuare la transazione evitando così di tenere aperta una transazione per troppo tempo.

Possiamo usare Service Broker per la raccolta di dati, ad esempio delle filiali di una banca potrebbero inviare le informazioni sulle transazioni al server centrale, che le processerà.

Oppure per grosse applicazioni che accedono a più database di SQL Server, come mezzo di interscambio di informazioni.

Service Broker può essere utilizzato per implementare dei sistemi di elaborazione query in modo affidabile, indipendentemente da eventuali errori interruzioni di servizio, inserendo i messaggi da elaborare in una coda. I messaggi saranno poi processati quando si avranno risorse disponibili.

O infine per operazioni di elaborazione batch su vasta scala. Un'applicazione potrebbe inviare i dati ad una coda di Service Broker mentre un'altra ne potrebbe leggere periodicamente il contenuto della coda per poi elaborare i dati.

ESEMPIO COMPLETO

Consideriamo il seguente scenario: una società con una serie di filiali ed una centrale.
Le filiali hanno il compito di registrare degli utenti sul database centrale. Le filiali comunicano con la centrale tramite Service Broker che

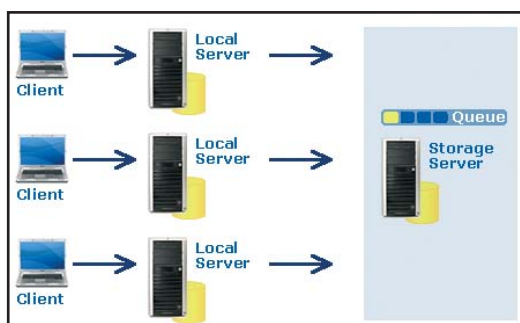



Fig. 7: *Il nostro scenario*

si occupa dell'accodamento delle richieste che verranno elaborate dal server.
Creiamo il database, impostiamo la proprietà TRUSTWORTHY a ON e creiamo infine una master key (è necessario per l'utilizzo di Service Broker)

```
CREATE DATABASE TestServiceBroker
USE TestServiceBroker

ALTER DATABASE TestServiceBroker SET TR
                                STWORTHY ON

CREATE MASTER KEY ENCRYPTION BY PASSWORD =
                            'MorpheusWeb2006'

Creiamo quindi la tabella utenti

CREATE TABLE TestServiceBroker.dbo.Utenti
(
        [idUtente] [int] IDENTITY(1,1) NOT NULL,
        [nome] [varchar](50) COLLATE Latin1_
                        General_CI_AS NOT NULL,
        [cognome] [varchar](50) COLLATE
                  Latin1_General_CI_AS NOT NULL,
        [email] [varchar](50) COLLATE Latin1_
                        General_CI_AS NOT NULL,
 CONSTRAINT [PK_Utenti] PRIMARY KEY CLUSTERED
(
        [idUtente] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

E due stored procedure che ci serviranno per inserire gli utenti

```
CREATE PROCEDURE dbo.getXMLValue
        @xmlString XML,
        @xpath varchar(100)          ,
    @value varchar(100)       output
AS
DECLARE @Idoc int
SET NOCOUNT ON
EXEC sp_xml_preparedocument @Idoc OUTPUT,
                                  @xmlString
SELECT  @value = [text] FROM   OPENXML (@Idoc,
                                  @xpath,1)
```

La prima legge un XML ed estrae il testo associato ad un elemento, mentre la seconda:

```
CREATE PROCEDURE dbo.ProcessaInformazioni
        @utente XML
AS

DECLARE @nomeUtente varchar(50)
EXECUTE dbo.getXMLValue @utente, 'Utente/nome',
                          @nomeUtente output
DECLARE @cognomeUtente varchar(50)
EXECUTE dbo.getXMLValue @utente,
         'Utente/cognome', @cognomeUtente output
DECLARE @emailUtente varchar(50)
EXECUTE dbo.getXMLValue @utente, 'Utente/email,
                          @emailUtente output
INSERT INTO Utenti(nome,cognome,email)
        VALUES(@nomeUtente,@cognome
                        Utente,@emailUtente)
```

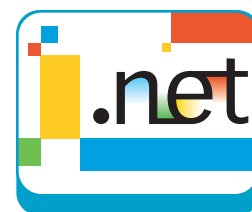
utilizza dbo.getXMLValue per leggere nome, cognome ed email dall'XML
La fase di setup del database è completata. Occorre adesso creare gli oggetti per lo scambio dei dati.
Come prima cosa creiamo uno schema per la validazione del messaggio XML

```
CREATE XML SCHEMA COLLECTION  SchemaUtente
AS
'<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Utente">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="cognome"
                             type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>'
```

Creiamo quindi in sequenza il message type ed il contract.

```
CREATE MESSAGE TYPE
  [tipoUtente]
VALIDATION =
        VALID_XML WITH SCHEMA COLLECTION
                             [SchemaUtente]
```

SQL 2005 EXPRESS EDITION

Chi utilizza SQL Server Express deve comunque avere almeno un'installazione di SQL Server 2005. Questo perché in base alla licenza d'uso non è possibile che un messaggio arrivi da un SQL Express ad un altro Express senza che almeno un altro SQL Server (non Express) lo abbia processato. Se ciò dovesse succedere il messaggio andrebbe perso.
Possiamo renderci conto di simili comportamenti utilizzando il Profiler; vedremmo dei messaggi del tipo: "This message has been dropped due to licensing restrictions".
In una situazione reale potremmo pensare di utilizzare delle copie della versione Express come Database per dei front-end ed una versione non Express per il back-end di elaborazione dati.

```
CREATE CONTRACT
[InserisciUtenteContract]
( [tipoUtente]  SENT BY ANY)
```

La seguente è la procedura che viene attivata quando ci sono messaggi in coda.

```
CREATE PROCEDURE dbo.doInserisciUtente
AS
BEGIN TRAN
 DECLARE @id uniqueidentifier
 DECLARE @utente XML
 --Leggo il messaggio
  WAITFOR
  (
    RECEIVE TOP(1) @id = conversation_handle,
                              @utente = message_body
    FROM ReceiverQueue
  )
 --processo il messaggio
 EXECUTE ProcessaInformazioni @utente
 --chiudo la conversation
 END CONVERSATION @id WITH CLEANUP
COMMIT TRAN
```

La Stored Procedure dbo.doInserisciUtente, legge il messaggio XML ed esegue ProcessaInformazioni (che inserisce l'utente nella tabella). Creiamo quindi la coda per i messaggi, indicando il nome della procedura da attivare alla ricezione.

```
CREATE QUEUE ReceiverQueue
  WITH STATUS = ON,
  ACTIVATION
  (
    PROCEDURE_NAME = doInserisciUtente,
    MAX_QUEUE_READERS = 5,
         EXECUTE AS SELF
  )
```

Ed il service

```
CREATE SERVICE [InserisciUtenteService]
  ON QUEUE ReceiverQueue([InserisciUtenteContract])
```

La fase di setup è completa.
Non ci resta che testare l'invio e la ricezione dei messaggi!
Creiamo il messaggio XML e lo inviamo alla coda

```
declare @utente XML
set @utente =  '<Utente>
                    <nome>Carmelo</nome>
                    <cognome>Scuderi</cognome>
<email>webmaster@morpheusweb.it</email>
                                        </Utente>'
```

```
DECLARE @dh uniqueidentifier
BEGIN DIALOG @dh
  FROM SERVICE [InserisciUtenteService]
  TO SERVICE  'InserisciUtenteService'
  ON CONTRACT  [InserisciUtenteContract];

--invia il messaggio
SEND ON CONVERSATION @dh
MESSAGE TYPE [tipoUtente] (@utente)
```

Per leggerlo dalla coda ed attivare la stored procedure che effettua l'inserimento:

```
RECEIVE TOP(1)
  CAST(message_body as XML)
  FROM ReceiverQueue
```

Il messaggio XML sarà processato dalla stored procedure che ne leggerà il contenuto ed inserirà i dati dell'utente in tabella.
Per semplicità di esposizione abbiamo lavorato su un unico database, ma è possibile utilizzare n database per l'invio dei messaggi ed un altro per la ricezione.

CONCLUSIONI

Grazie a Service Broker è possibile implementare delle architetture Service Oriented, in cui le applicazioni inviano messaggi a SQL Server e continuano ad eseguire i compiti che stavano svolgendo. Questo consente di progettare soluzioni robuste e scalabili con uno sforzo trascurabile da parte del programmatore che non deve più preoccuparsi della gestione di complesse strutture dati per l'accodamento delle richieste o la verifica del recapito dei messaggi.

Carmelo Scuderi

L'AUTORE

Carmelo Scuderi è ingegnere informatico. Si occupa di sviluppo software web-based per una società di telecomunicazioni di Milano. Gestisce un sito web ricco di script e manuali per chi si affaccia al mondo della programmazione web (www.morpheusweb.it)

ABILITARE E DISABILITARE SERVICE BROKER

Service Broker è attivato di default quando viene creato un database.
È comunque possibile abilitarlo o disabilitarlo tramite delle semplici query T-SQL.

Per disabilitarlo

```
ALTER DATABASE nome_database
SET DISABLE_BROKER
```

Per abilitarlo

```
ALTER DATABASE nome_database
SET ENABLE_BROKER
```

È poi possibile visualizzare lo stato di attivazione di Service Broker tramite una view di sistema sys.databases:

```
SELECT
  service_broker_guid,
  is_broker_enabled
FROM
  sys.databases
WHERE
  [name] = 'nome_database';
```

MENU GRAFICI E A SCOMPARSA!

JAVA SI EVOLVE, ED È ORA POSSIBILE CREARE APPLICAZIONI CHE FANNO USO DI MENU COMPLESSI, FORMATI DA UNA MISCELA DI GRAFICA, TESTO ED ELEMENTI MULTIMEDIALI. INOLTRE SI POSSONO REALIZZARE ALCUNI SIMPATICI EFFETTI...

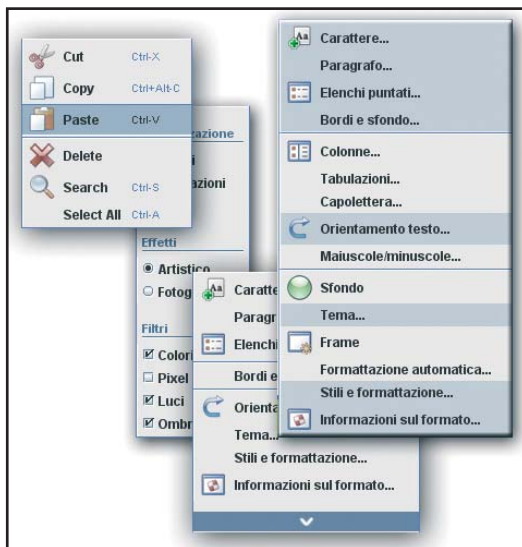



Fig. 1: *Un esempio dei menu grafici che intendiamo ottenere utilizzando swing*

Creare, aprire o chiudere un documento sono tutte operazioni che effettuiamo comunemente. I menu ci aiutano a scegliere i giusti comandi. Essi offrono una breve descrizione di tutto ciò che si può fare all'interno di un sistema. Arricchire ed ampliare un menu significa quasi sempre aumentare l'usabilità del software. Nel corso dell'articolo vedremo come Java è in grado di costruire e personalizzare un menu.

MUOVIAMO I PRIMI PASSI

La libreria grafica AWT, presente fin dalle prime versioni di Java, contiene alcune classi per l'implementazione dei menu. Tali classi furono progettate in modo da funzionare in maniera autonoma. In parole povere "disegnavano" loro stesse. Con l'introduzione delle librerie Swing, il framework è stato reso più modulare e scalare. Ogni oggetto grafico estende la classe Component, responsabile della gestione degli eventi e delle operazioni di rendering a basso livello. Vediamo in dettaglio come creare una barra dei menu e quali sono le differenze sostanziali per chi utilizza ancora le vecchie classi.

APPROCCIO AWT

Il metodo *setMenuBar(MenuBar mb)*, appartenente alla classe Frame, visualizza una barra dei menu nella parte alta della finestra di applicazione. Ogni singolo menu fa riferimento ad oggetti dell'omonima classe Menu, costruiti con un'etichetta di testo. Creiamo una barra dei menu con due voci e aggiungiamola al Frame corrente:

```
Menu menuFile = new Menu("File");
Menu menuModifica = new Menu("Modifica");
MenuBar mb = new MenuBar();
mb.add(menuFile);
mb.add(menuModifica);
setMenuBar(mb);
```

Costruire un menu significa costruire una struttura gerarchica, in particolare un albero, con nodi intermedi e nodi foglia. Infatti un menu può contenere al suo interno due tipologie di oggetti: una singola voce (nodo foglia) o un sottomenu (nodo intermedio). Ogni sottomenu è a sua volta un oggetto della classe Menu, contenente nodi foglia ed eventuali nodi intermedi. Le singole voci non sono altro che oggetti della classe MenuItem. Vediamo un esempio:

```
MenuItem menuFileNuovo1 = new
                    MenuItem("Documento");
MenuItem menuFileNuovo2 = new
                    MenuItem("Archivio");
Menu menuFileNuovo = new Menu("Nuovo");
menuFileNuovo.add(menuFileNuovo1);
```

```
menuFileNuovo.add(menuFileNuovo2);
MenuItem menuFileEsci = new MenuItem("Esci");
menuFile.add(menuFileNuovo);
menuFile.add(menuFileEsci);
```

Abbiamo utilizzato il solo metodo *add (MenuItem mi)* per inserire all'interno del menu File sia un sottomenu sia una singola voce. Un'occhiata alle API Java ci farà notare che la classe Menu estende la classe *MenuItem*.
L'utilizzo delle classi AWT non ci consente di andare oltre i semplici menu di testo. L'inserimento di bottoni ed immagini è possibile solo grazie alle estensioni Swing.

APPROCCIO SWING

Quanto detto si può riadattare per le classi Swing. Teniamo presente però che le classi *JMenuBar*, *JMenu* e *JMenuItem* non funzionano in maniera autonoma dal punto di vista grafico, ma estendono tutte *JComponent*. In particolare, *JMenu* e *JMenuItem* sono un'estensione di *AbstractButton*. Creiamo alcuni elementi ed inseriamoli all'interno di un menu:

```
JMenuItem menuFileCreateJTextArea = new
                    JMenuItem("JTestArea");
JMenuItem menuFileCreateJLabel = new
                    JMenuItem("JLabel");
JMenuItem menuFileCreateJComboBox= new
                    JMenuItem("JComboBox");
JMenu menuFileCreate = new JMenu("Create");
menuFileCreate.add(menuFileCreateJLabel);
menuFileCreate.add(menuFileCreateJTextArea);
menuFileCreate.add(menuFileCreateJComboBox);
```

Aggiungiamo il menu appena costruito all'interno del menu principale "File", inserendo eventuali elementi sparsi.

```
JMenuItem menuFileOpen = new
                    JMenuItem("Open");
JMenuItem menuFileSave = new JMenuItem("Save");
JMenuItem menuFileSaveAs = new
                    JMenuItem("Save as...");
JMenuItem menuQuit = new JMenuItem("Quit");
JMenu menuFile = new JMenu("File");
menuFile.add(menuFileCreate);
menuFile.addSeparator();
menuFile.add(menuFileOpen);
menuFile.add(menuFileSave);
menuFile.add(menuFileSaveAs);
menuFile.addSeparator();
menuFile.add(menuQuit);
```

Visualizziamo il tutto creando una nuova *JMenuBar* e associandola al *JFrame* corrente:

```
JMenuBar menuBar = new
                    JMenuBar();
menuBar.add(menuFile);
setJMenuBar(menuBar);
```

Nell'esempio i vari elementi sono raggruppati attraverso delle linee separatrici, generate dal metodo addSeparator(). In figura è mostrato il risultato.

Fig. 2: ***In sottomenu ed alcuni elementi aggiuntivi. La grafica produce un impatto visivo carente.***

Compilando ed eseguendo il codice precedente ci accorgiamo che un click del mouse sopra una voce di menu non produce alcun effetto. Dobbiamo quindi, proprio come nel caso di un *JButton*, aggiungere un oggetto ascoltatore presso *JMenuItem*, in modo da catturarne gli eventi associati. Il modo migliore per farlo è quello di implementare un Action listner. Il codice seguente è un esempio di come possiamo fare

```
menuFileQuit.addActionListener(new ActionListener()
                    {
public void actionPerformed(ActionEvent e){
                    System.exit(0);
}});
```

IL MENU DI POPUP

È un oggetto largamente utilizzato in tutti i software. In genere si tratta del menu che compare quando facciamo un click destro con il mouse. È facile da creare perché segue le stesse regole valide per i menu, ed in più può essere associata a qualunque componente all'interno dell'applicazione. È un utile strumento per eseguire comandi, in determinati contesti, in maniera semplice e veloce.

IMMAGINI E SCORCIATOIE

In genere nelle comuni applicazioni vediamo piccole icone accanto ad alcune voci di menu. Le immagini non hanno solo una funzione decorativa ma consentono all'utente di riconoscere "a colpo d'occhio" il comportamento del comando associato. Anche con Java è possibile fare ciò. Un oggetto di classe *JMenuItem* può essere costruito a partire da una stringa di testo e da un'immagine:

```
JMenuItem menuEditCopy = new JMenuItem("Copy",
                    new ImageIcon("copy.png"));
```

Altra pratica comune è quella di aggiungere ad ogni voce di menu una scorciatoia da tastiera. Ecco come:

```
menuEditCopy.setAccelerator(KeyStroke.getKeyStro
        e(KeyEvent.VK_C, InputEvent.CTRL_MASK));
```

Il secondo parametro del metodo *getKeyStroke*() è una maschera di bit composta dalla somma logica delle varie costanti contenute nella classe InputEvent. Se volessimo associare, ad esempio, alla precedente voce di menu la combinazione di tasti CRTL+ALT+C dovremmo fare così:

```
menuEditCopy.setAccelerator(KeyStroke.getKeyStro
        e(KeyEvent.VK_C, InputEvent.CTRL_MASK +
                    InputEvent.ALT_MASK));
```

Applicando le precedenti migliorie il menu

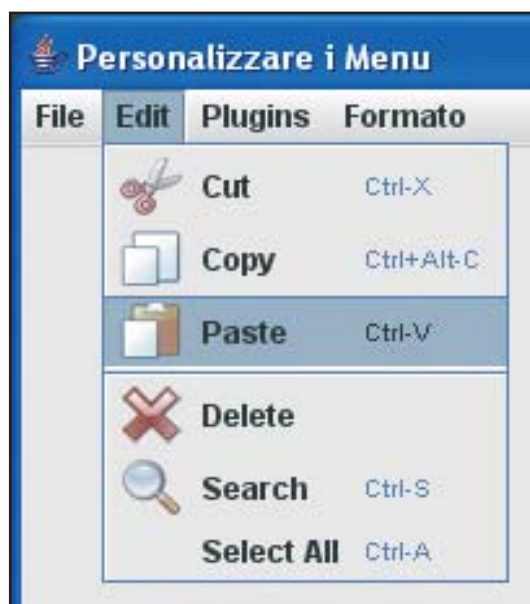



Fig. 3: *In figura si può apprezzare l'estetica del menu una volta inserite le immagini. Le scorciatoie da tastiera, di diverso colore e dimensione, sono posizionate alla sinistra del testo.*

acquisterà più eleganza e sarà un pizzico più funzionale. Spingiamoci oltre e miglioriamo il raggruppamento degli elementi.

GRUPPI ESPLICATIVI

Il raggruppamento delle voci di menu tramite l'inserimento di semplici linee separatrici ha un impatto puramente visuale. Esso non dice nulla riguardo le categorie di azioni in esso contenute. Associare un'etichetta di testo per ogni gruppo di un menu sta diventando una pratica sempre più utilizzata. L'utente riconosce così con più accortezza i comandi. In Java possiamo realizzare tale caratteristica aggiungendo un separatore personalizzato direttamente alla classe *JMenu*. Creiamo la classe *JMenuSeparator*, estensione di *JPanel*, con il seguente costruttore:

```
public JMenuSeparator(String title) {
    setLayout(new BorderLayout());
    JLabel label = new JLabel(title);
    JSeparator separator = new
                    JSeparator();
    label.setForeground(new
                Color(90,121,160));
    label.setFont(new
  Font(label.getFont().getName(),Font.BOLD,11));
    separator.setForeground(new
                Color(90,121,160));
    add(label, "Center");
    add(separator, "South");
setBorder(BorderFactory.createEmptyBorder(15,5,5,
                                        ));
}
```

Istanziamo un nuovo oggetto JMenu Separator per ogni gruppo e posizioniamolo in cima agli oggetti che rappresenta. Le classi JCheck BoxMenuItem e JRadioButton MenuItem sono un modo veloce per inserire caselle di spunta all'interno dei menu.

```
JRadioButtonMenuItem
        menuPluginsArtistico = new
    JRadioButtonMenuItem("Artistico", true);
JRadioButtonMenuItem
        menuPluginsFotografico = new
    JRadioButtonMenuItem("Fotografico");
ButtonGroup group = new
                ButtonGroup();
group.add(menuPluginsArtistico);
group.add(menuPluginsFotografico);
JCheckBoxMenuItem
        menuPluginsColori = new
    JCheckBoxMenuItem("Colori", true);
```

```
JCheckBoxMenuItem
        menuPluginsPixel = new
        JCheckBoxMenuItem("Pixel");
JCheckBoxMenuItem menuPluginsLuci = new
        JCheckBoxMenuItem("Luci",true);
JCheckBoxMenuItem menuPluginsOmbre = new
        JCheckBoxMenuItem("Ombre", true);
JMenu menuPlugins = new JMenu("Plugins");
menuPlugins.add(new JMenuSeparator("Effetti"));
menuPlugins.add(menuPluginsArtistico);
menuPlugins.add(menuPluginsFotografico);
menuPlugins.add(new JMenuSeparator("Filtri"));
menuPlugins.add(menuPluginsColori);
menuPlugins.add(menuPluginsPixel);
menuPlugins.add(menuPluginsLuci);
menuPlugins.add(menuPluginsOmbre);
```

Fig. 4: ***E' decisamente meglio cercare qualcosa all'interno di un menu raggruppato, specie se si tratta di spuntare delle CheckBox.***

MENU FANTASMA

Sebbene il nome può suonare strano, abbiamo avuto a che fare un po' tutti noi con questi particolari menu. Basta utilizzare le applicazioni contenute nelle ultime versioni del pacchetto Microsoft Office. In pratica, ad un primo clic del mouse vengono visualizzate solo quelle voci di menu che, secondo il progettista, risultano più importanti per l'utente. Un secondo clic del mouse espande invece il menu in maniera completa. Vediamo come incorporare questo interessante effetto all'interno di un'applicazione Java.

Per prima cosa estendiamo la classe *JMenuItem* includendo una variabile booleana per ricordarci se l'elemento è visibile oppure no. Chiamiamo questa nuova classe *ExtJMenuItem*. Includiamo anche metodi per l'information hiding:

```
public class ExtJMenuItem extends JMenuItem {
    protected boolean extvisible  = true;
    public ExtJMenuItem(String text, boolean
                                    visible) {
        super(text);
        this.extvisible = visible;
    }
    public ExtJMenuItem(String text, boolean
                          visible, Icon icon) {
        super(text, icon);
        this.extvisible = visible;
    }
    public void setExtVisible(boolean visible) {
        this.extvisible = visible;
    }
    public boolean isExtVisible() {
        return extvisible;
    }
}
```

Per incorporare la nuova funzionalità abbiamo la necessità di estendere la classe JMenu. Teniamo traccia in un vettore dei nuovi elementi aggiunti. Poi aggiorniamo i cambiamenti.

```
    public void add(ExtJMenuItem menuItem) {
        items.add(menuItem);
        refresh();
    }
```

Il metodo *refresh*() riposiziona gli elementi visualizzando solo quelli per cui la variabile extvisible è pari a true. Al termine della procedura è necessario un refresh grafico:

```
    public void refresh() {
        ...
        updateUI();
        getPopupMenu().pack();
}
```

Come ultimo elemento aggiungiamo un bottone per espandere l'intero menu. Esso avrà associato il seguente ascoltatore:

MENU E HCI

HCI è acronimo di Human Computer Interaction. Si tratta di una scienza che studia l'usabilità dei sistemi e l'approccio uomo-macchina. Secondo alcuni studi, i menu non dovrebbero essere né troppo lunghi né troppo profondi, in quanto causa di disorientamento nell'utente. Un valore massimo di profondità si aggira intorno ai 7 livelli. In sostanza i menu non devono avere una struttura molto complessa.

```
new ActionListener() {
        public void
        actionPerformed(ActionEvent e) {
                showAll();
        }
}
```

showAll() contiene una routine simile a quella di refresh(), visualizzando proprio tutti gli elementi del vettore items. In futuro, quando il menu verrà riaperto, dovrà presentarsi col suo stato iniziale. All'interno del costruttore relativo a ExtJMenu, aggiungiamo un ascoltatore che richiami il metodo di aggiornamento refresh():

```
addMenuListener(new
        MenuListener() {
        public void
        menuCanceled(MenuEvent me) {}
        public void
        menuSelected(MenuEvent me) { refresh(); }
        public void
        menuDeselected(MenuEvent me) {}
}
);
```

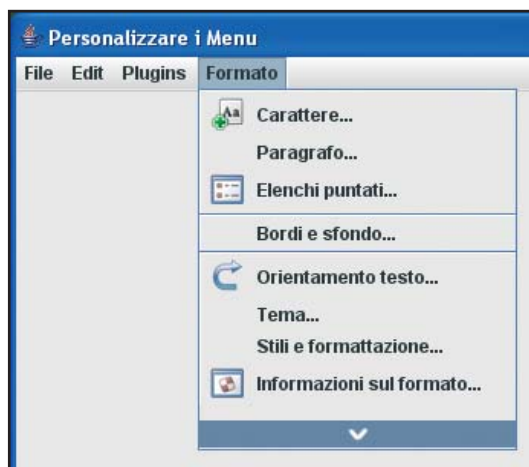



Fig. 5: ***Ecco il risultato finale. Cliccando sul bottone con la freccia, ultimo elemento del menu, vengono visualizzate tutti gli elementi nascosti.***

MENU CIRCOLARI

I classici menu che tutti conosciamo non sono l'unico modo per raggruppare i comandi. In passato fu concepito anche il concetto di menu circolare. Gli elementi sono distribuiti ad una stessa distanza da un punto fisso. Anziché scorrere un intero menu, l'occhio umano deve scostarsi solo di poco per raggiungere un certo obiettivo. Ne consegue però un limitato spazio utilizzabile. Se i vecchi menu sono ancora largamente utilizzati, ciò significa che non tutte le nuove tecnologie sono allo stesso tempo innovative.

Fig. 6: ***Una volta che il menu è stato espanso, gli elementi principali possiedono un diverso colore di sfondo. L'aggiunta delle immagini rende la grafica gradevole ed esplicativa.***

CONCLUSIONI

Abbiamo sfruttato solo alcune tecniche per rendere i nostri menu più usabili e gradevoli esteticamente.
Colori, forme e caratteri possono essere combinati in un'infinita varietà di modi e stili. Swing in questo senso rappresenta una via d'uscita molto efficace per coloro che vogliono creare applicazioni moderne e funzionali. Non c'è un reale appesantimeto delle prestazioni, e per contro l'usabili à delle applicazioni viene enormemente aumentata.
Nel progettare i vostri software tenete sempre conto che l'utilizzatore finale è spesso qualcuno che non ha una grande esperienza di informatica, per tale motivo nella progettazione per quanto possa sembrare frivolo è importante concentrarsi su tutte quelle estensioni che possono rendenre il software più fruibile. Un enorme peso in questo lo assumono i menu che rappresentano sicuramente lo strumento più utilizzato dall'u ente finale. La funzione delle icone in una voce di menu è quella di assolvere ad un impatto visivo che non è esclusivamente estetico ma che anzi aumenta la comprensione di una funzione da parte di chi utilizza un programma. In conclusione per realizzare applicazioni professionali è importante anche concentrarsi su questi strumenti

Antonio Trapani

INTEGRIAMO LE MAPPE SUL POCKET PC

IN QUESTO ARTICOLO SFRUTTEREMO UN WEB SERVICE DI MICROSOFT PER VISUALIZZARE SU UNA MAPPA LA POSIZIONE DEGLI INDIRIZZI SALVATI SULLA RUBRICA. CON VISUAL STUDIO 2005 MOLTE OPERAZIONI RISULTERANNO SEMPLIFICATE...

Nel precedente articolo, abbiamo visto come realizzare un'applicazione completa che permette, agli agenti di una società, di creare ordini su un dispositivo mobile. Per farlo, abbiamo usato Microsoft .NET Compact Framework 2.0 e un dispositivo Windows Mobile 5.0. Nell'articolo abbiamo visto come questi due prodotti, rendano molto semplice lo sviluppo di applicazioni anche complesse, grazie tanto alle novità introdotte dal Compact Framework quanto dalle nuove API esposte dal nuovo sistema operativo.

Fig. 1: *L'applicazione che realizzeremo*

In questo articolo, ci concentreremo sull'aggiunta di una funzionalità particolare al nostro software. Sebbene non sia di vitale importanza, è decisamente comodo per una agente visualizzare la mappa del luogo in cui si trova il cliente. Vedremo quindi come farlo prendendo direttamente i dati dall'anagrafica ed usando il servizio web messo a disposizione da MapPoint.
Iniziamo!

MAPPOINT

MapPoint è il sistema sviluppato da Microsoft per la localizzazione geografica. Questo prodotto è distribuito sostanzialmente in 3 versioni diverse adatte a scopi specifici. Vediamole brevemente:

- **MapPoint Web Service:** è un servizio web che espone sostanzialmente il motore di MapPoint. Attraverso l'utilizzo di questo servizio è possibile realizzare applicazioni personalizzate per la gestione della localizzazione geografica senza preoccuparsi di istallare altri prodotti sui PC dei nostri clienti.
- **MapPoint Location Server:** è un sistema di localizzazione real time che sfrutta dispositivi mobili e MapPoint Web Service per la localizzazione immediata di entità in movimento.
- **MapPoint 2004:** è la versione software di MapPoint. Questa versione è stata realizzata sia per poter essere utilizzata direttamente, sia per poter visualizzare informazioni provenienti da file del pacchetto office.

Escludendo la versione *MapPoint Location Server* (la cui implementazione esula dallo scopo di questo articolo), per una applicazione standard abbiamo la possibilità di scegliere tra la versione *MapPoint Web Service* e la versione *MapPoint 2004*. Tale scelta deve essere fatta in modo ponderato onde evitare di sprecare i nostri soldi (o quelli dei nostri clienti). I costi delle 2 versioni sono abbastanza diversi tra loro, sebbene quello della versione client (*MapPoint 2004*) è definito ($ 299) e quello della versione WebService sia abbastanza complesso da calcolare e bisogna contattare Microsoft per avere una stima precisa dei costi.
Il mio personale consiglio è:

- **MapPoint 2004:** per singole applicazioni, non multiutenza e che devono lavorare in modalità disconnessa.
- **MapPoint Web Service:** per applicazioni

Impegno

Tempo di realizzazione

client-server, applicazioni web, installazioni multiple, applicazioni connesse.

Per applicazioni per dispositivi mobili invece, il problema della scelta non si pone: dobbiamo usare il servizio web.

IL SERVIZIO WEB DI MAPPOINT

Il servizio web di MapPoint permette agli sviluppatori di creare applicazioni basate sul motore di generazione della mappe, sfruttando appunto un servizio web.
I vantaggi derivanti dall'utilizzo di questo sistema sono molteplici. Innanzitutto non c'è la necessità di installare componenti sui PC. Il servizio web infatti, permette di eseguire quasi tutte le operazioni possibili con la versione client senza la necessità che essa sia installata. Altro vantaggio è sicuramente la semplicità di utilizzo, soprattutto in applicazione basate sul Microsoft .NET Framework. Vedremo infatti che, con poche righe di codice, sarà possibile generare mappe, semplicemente passando al servizio web un indirizzo.
Uno degli svantaggi intrinseci di questa tecnologia è però la necessità di un collegamento ad internet. In servizio web infatti, come dice il termine stesso, scambia informazioni con la nostra applicazione attraverso il web. Motivo per cui, il vincolo di utilizzo di questa tecnologia è avere una connesione attiva.
Prima di iniziare ad utilizzare il servizio web, è necessario compiere alcune operazioni preliminari.
Innanzi tutto il servizio non è pubblico ma privato ed a pagamento. Fortunatamente, Microsoft mette a disposizione di noi sviluppatori degli account di test gratuiti. Sostanzialmente ci sono due differenze tra l'account di staging e quello reale: l'account di staging consente un massimo di mille interrogazioni al giorno (più che sifficienti in fase di sviluppo) a differenza di quello a pagamento in cui non c'è questo limite. La seconda differenza è che il servizio a pagamento è tendenzialmente più veloce di quello di staging.
Questi due limiti però, non complicano in alcun modo lo sviluppo ed il test delle nostre applicazioni.
Richiedere l'account di staging è abbastanza semplice: basta accedere al seguente indirizzo web: *https://mappoint-css.partners.extranet.microsoft.com/MwsSignup/Eval2.aspx*, compilare il form con la richiesta ed attendere un paio di giorni che venga attivato.

Una volta ricevute le password per accedere al servizio, siamo pronti ad iniziare lo sviluppo.

PREDISPOSIZIONE DELL'AMBIENTE

Ricevute le credenziali di accesso al servizio web, dobbiamo predisporre l'ambiente di sviluppo e soprattutto l'emulatore per il corretto funzionamento.
Il primo passo è quello di installare, se non lo abbiamo già fatto, il Windows Mobile 5.0 SDK (vedi box laterale) e configurare il nostro emulatore ad usare la connessione ad internet del PC su cui stiamo lavorando.
Per farlo, avviamo l'emulatore da Microsoft Visual Studio .NET 2005, selezioniamo il menu File e la voce Configura. Dovrebbe apparire una maschera simile a quella di **Figura 2**.

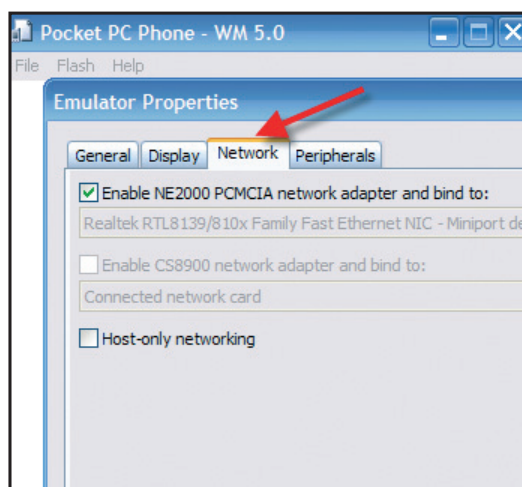



Fig. 2: ***La configurazione dell'emulatore***

Scegliamo quindi la scheda Network e spuntiamo la voce *"Enable NE2000 PCMCIA Network Adapter and bind to:"* scegliendo, ove possibile, una scheda di rete del nostro PC. L'operazione appena eseguita permetterà all'emulatore di collegarsi alla nostra rete e di navigare sul web.
Per verificare che tutto sia configurato a dovere, apriamo il *Pocket Internet Explorer* e proviamo a navigare su un sito.
Completata questa operazione, non ci resta che scrivere il nostro codice.

LA NOSTRA APPLICAZIONE

Come applicazione d'esempio, usiamo la stessa vista nell'articolo precedente, a cui ov-

Tutti i dettagli relativi a MapPoint possono essere recuperati dal sito ufficiale al seguente indirizzo: http://www.microsoft.com/mappoint/default.mspx. **Seguendo i link relativi al servizio web sarà poi possibile attivare un account di tipo Developer per eseguire i propri test.**

viamente aggiungeremo gli elementi utili a generare e visualizzare le mappe.
Avendo a che fare con un dispositivo mobile, sappiamo già di avere a disposizione poco spazio sul display. Ne consegue che la mappa dovrà essere visualizzata in un nuvo form. Aggiungiamone quindi uno al progetto che chiameremo *MapRender.cs*. Al suo interno inseriremo una picture box *(pictureBoxMappa)*, il cui scopo sarà quello di visualizzare la nostra mappa ed una label *(labelIndirizzoCliente)* che mostrerà l'indirizzo del cliente selezionato. Il nuovo form dovrebbe apparire come in **Figura 3**.

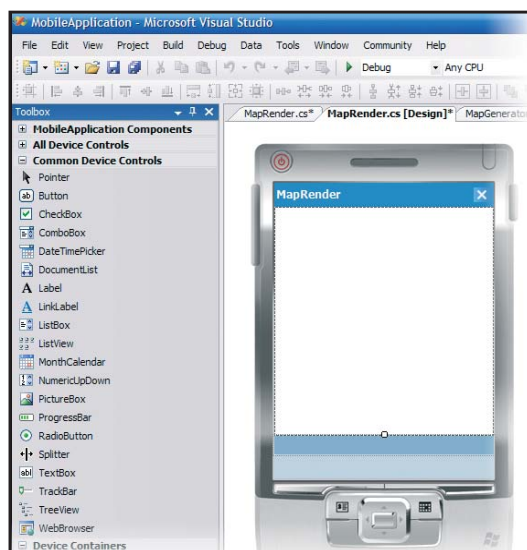



Fig. 3: *Il form per la visualizzazione delle mappe*

Modifichiamo ora il costruttore del form in modo che accetti dei parameri in ingresso. Quello che vogliamo infatti è che, una volta aperto, il form mostri direttamente la mappa relativa all'indirizzo di un cliente. Per farlo, l'indirizzo deve essere passato al form:

```
namespace MobileApplication {
    public partial class MapRender : Form {
        public MapRender(string Cliente, string
                    indirizzo, string cap, string citta) {
            InitializeComponent();
            if ( Cursor.Current == Cursors.WaitCursor){
                Cursor.Current = Cursors.Default;
            }
            this.Name = "Cliente: " + Cliente;
            labelIndirizzoCliente.Text = indirizzo + " "
                                    + cap + ", " + citta;
        }
    }
}
```

Dopo l'inizializzazione dei componenti *(InitializaComponent())*, impostiamo anche il testo della label *labelIndirizzoCliente* con i dati presi dal costruttore.
Pronto il form, non dobbiamo aprirlo, e lo facciamo aggiungendo un nuovo bottone *(btnFindMap)* nel form in cui sono presenti le anagrafiche dei clienti *(Clienti.cs)* come in **Figura 4**.

Fig. 4: *L'aggiunta del bottonre Mappa al form dei clienti*

Il codice da associare all'evento *click* di questo bottone servirà quindi ad aprire il form *MapRender.cs* passandogli i parametri corretti:

```
private void btnFindMap_Click(object sender,
                                        EventArgs e) {
    Cursor.Current = Cursors.WaitCursor;
    string Cliente = string.Empty;
    int indiceRiga;
    indiceRiga =
            dataGridClienti.CurrentCell.RowNumber;
    Cliente = appDatabaseDataSet.tblClienti.Rows[
                            indiceRiga][1].ToString();
    MapRender mapRender = new MapRender(Cliente,
        lblIndirizzo.Text, lblCAP.Text, lblCittà.Text);
    mapRender.ShowDialog();
}
```

Oltre ai dati relativi all'indirizzo, ci conviene passare il nominativo del cliente. Trovandosi in un form separato infatti, il nostro utilizzatore potrebbe non essere sicuro di aver cliccato sul cliente corretto quindi è il caso di visualizzarlo. Sebbene il nominativo del cliente si trova all'interno della *DataGridView* e potrebbe essere recuperato da lì, converrebbe recuperarlo dal *DataSource*. Le colonne della griglia possono infatti essere modificate, nascoste o spostarte e si correrebbe il rischio di recuperare un dato errato.
Ora che abbiamo il form pronto ed i parametri con cui generare la mappa, siamo pronti ad usare il servizio web di *MapPoint*.
Dopo aver creato il riferimento web, aggiungiamo al nostro progetto una nuova classe denominata *MapGenerator.cs* che avrà il compi-

SUL WEB

DOVE TROVO IL WINDOWS MOBILE 5.0 SDK?

Il Microsoft Windows Mobile 5.0 SDK è scaricabile al seguente indirizzo web: http://www.microsoft.com/downloads/details.aspx?familyid=83A52AF2-F524-4EC5-9155-717CBE5D25ED&displaylang=en ed include gli emulatori ed i template di Microsoft Visual Studio .NET 2005.

to di generare la mappa e restituirla sotto forma di immagine.
Una delle prime cose da fare nella nostra classe è quella di aggiungere la direttiva *"using MobileApplication.net.mappoint.staging;"* in modo da poter usare i metodi del servizio web in modo più comodo.
Come detto in precedenza, al fine di poter generare una mappa, abbiamo bisogno di alcuni elementi. Un punto su una mappa può infatti essere localizzato in due modi: o attraverso una coppia di coordinate geografiche (latitudine e longitudine) di cui però non disponiamo, o attraverso un indirizzo (che abbiamo). Questo però, deve essere prima "trovato" da MapPoint, e solo successivamente potrà essere usato per disegnare la mappa.
Dopo aver passato al servizio web le nostre credenziali con il codice:

```
findService.Credentials = new
        System.Net.NetworkCredential(UserName,
                                     Password);
findService.PreAuthenticate = true;
renderService.Credentials = new
        System.Net.NetworkCredential(
                          UserName, Password);
renderService.PreAuthenticate = true;
```

sia al servizio di ricerca *(findService)* che a quello di generazione della mappa *(renderService)*, eseguiamo una ricerca all'interno del Data Base di MapPoint con il seguente codice:

```
//Creiamo un oggetto Address usando l'indirizzo del
                                            cliente
Address myAddress = new Address();
myAddress.AddressLine = Indirizzo;
myAddress.PrimaryCity = Citta;
myAddress.PostalCode = CAP;
myAddress.CountryRegion = "IT";

//Specifichiamo in quale Data Source l'indirizzo deve
                                    essere ricercato
findAddressSpec.DataSourceName = "MapPoint.EU";
findAddressSpec.InputAddress = myAddress;

//Eseguiamo la ricerca dell'indirizzo
foundAddressResults = findService.FindAddress(
                                  findAddressSpec);
```

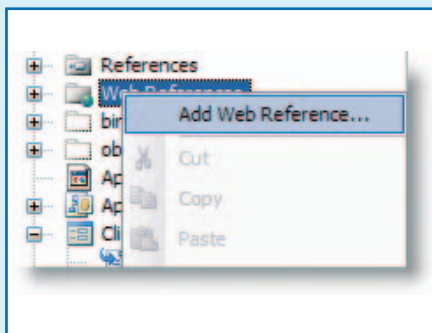
Dopo aver localizzato l'indirizzo, dobbiamo specificare a MapPoint "cosa" vogliamo vedere. MapPoint infatti, contiene le mappe di quasi tutto il pianeta e a noi serve solo una piccola porzione di esso. Per farlo, usiamo l'oggetto *ViewByScale* che ci permette di definire una scala per la nostra mappa:

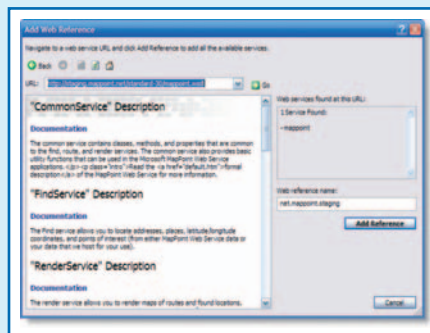
```
views = new ViewByScale[1];
views[0] = new ViewByScale();
views[0].CenterPoint = new LatLong();
```

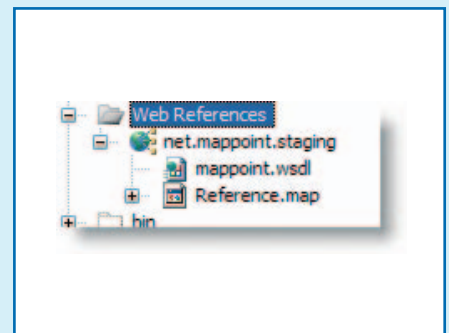
Uno dei siti di riferimento per chi deve implementare soluzioni basate su MapPoint è sicuramente http://www.mp2kmag.com. In esso si trovano numerosi articoli, tips, spezzoni di codice che spiegano in modo preciso i passi da seguire per implementare soluzioni basate su questo prodotto.

UTILIZZIAMO IL WEB SERVICE

Per utilizzare il servizio web di MapPoint, dopo aver ricevuto da Microsoft le credenziali di accesso, dobbiamo aggiungere un riferimento web al nostro progetto. Le istruzioni sono riportare nel riquadro in basso.

> IL MENU ADD REFERENCE

1 Facciamo click con il tasto destro del mouse sulla cartella *"Web Reference"* del nostro progetto nel solution explorer. Selezioniamo la voce *"Add web reference"* Questa operazione ci consentirà di accedere ad una seconda dialog box in cui potremo indicare la posizione del Web Service

> IL WEB SERVICE

2 Nella maschera che verrà aperta, imminseriamo l'indirizzo del servizio web (http://staging.mappoint.net/standard-30/mappoint.wsdl) e clicchiamo sulla freccia verde. Verranno visualizzati tutti i metodi che il servizio espone ed il nome da assegnare al riferimento web (net.mappoint.staging)

> METODI E PROPRIETÀ

3 Da questo istante sarà possibile usare tutti i metodi esposti dal servizio come se fossero relativi ad un oggetto locale. L'aggiunta del riferimento ha infatti creato una classe denominata proxy che ha lo scopo di rendere trasparente, per lo sviluppatore, il web service

```
views[0].CenterPoint.Latitude =
                    foundAddressResults.Results[0]
                    .FoundLocation.LatLong.Latitude;
views[0].CenterPoint.Longitude =
                    foundAddressResults.Results[0
                    ].FoundLocation.LatLong.Longitude;
views[0].MapScale = 250000;
```

All'istanza di *ViewByScale* specifichiamo una serie di parametri come il centro della mappa e la scala.
Siamo quasi pronti a mostrare il risultato. Ci mancano però ancora due elementi.
Il primo è la famosa bandierina che, sulla mappa, indica il punto preciso da noi ricercato. In MapPoint si chiama *PushPin* che definiamo come segue:

```
Pushpin[] pushpins = new Pushpin[1];
pushpins[0] = new Pushpin();
pushpins[0].IconDataSource = "MapPoint.Icons";
pushpins[0].IconName = "0";
pushpins[0].LatLong = views[0].CenterPoint;
pushpins[0].ReturnsHotArea = true;
pushpins[0].Label = cliente;
```

Per la definizione della nostra bandierina, specifichiamo un insieme di icone (un elenco completo è disponibile al seguente indirizzo: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mappointsdk/html/Tables-Icons.asp*) , il punto preciso in cui deve essere disegnata ed una label, che nel nostro caso sarà il nome del cliente.
Il secondo elemento di cui abbiamo bisogno è una serie di specifiche per il disegno della mappa. Se fino ad ora abbiamo trovato il punto, specificato quale parte del pianeta vogliamo vedere e dove puntare la "bandierina", ora dobbiamo mettere insieme tutti questi elementi nell'oggetto *MapSpecification:*

```
MapSpecification mapSpec = new MapSpecification();
mapSpec.DataSourceName = "MapPoint.EU";
mapSpec.Views = views;
mapSpec.Pushpins = pushpins;
mapSpec.Options = new MapOptions();
mapSpec.Options.Format = new ImageFormat();
mapSpec.Options.Format.Width = x;
mapSpec.Options.Format.Height = y;
mapSpec.Options.Style = MapStyle.Political;
```

Ora abbiamo tutti gli elementi e possiamo chiedere al servizio web di mostrarci la mappa:

```
MapImage[] mapImages =
                    renderService.GetMap(mapSpec);
```

```
System.IO.Stream streamImage = new
System.IO.MemoryStream(
                    mapImages[0].MimeData.Bits);
Bitmap bitmapImage = new Bitmap(streamImage);
return bitmapImage;
```

Per usarla, non ci resta che completare il codice del costruttore del form *MapRender.cs* con le seguenti istruzioni:

```
namespace MobileApplication {
    public partial class MapRender : Form {
        public MapRender(string Cliente, string
                    indirizzo, string cap, string citta) {
            InitializeComponent();
            if ( Cursor.Current == Cursors.WaitCursor)
            {
                Cursor.Current = Cursors.Default;
            }
            this.Name = "Cliente: " + Cliente;
            labelIndirizzoCliente.Text = indirizzo + " "
                                    + cap + ", " + citta;
            MapGenerator mp = new MapGenerator();
            this.pictureBoxMappa.Image =
                mp.MakeMap(Cliente, indirizzo, cap, citta,
                                    pictureBoxMappa.Width,
                                    pictureBoxMappa.Height);
        }
    }
}
```

Il risultato sarà quello visibile in **Figura 1**.

CONCLUSIONI

I dispositivi mobili hanno cambiato il nostro modo di lavorare rendendo il nostro lavoro decisamente più comodo. Se provassimo a pensare a come avrebbe dovuto lavorare il nostro agente solo cinque anni fa (carta, penna, tuttocittà, cartina geografica, agendina etc.), ci potremo rendere subito conto delle potenzialità fornite dalla tencologia. Per quanto riguarda la nostra professione, i nuovi strumenti, i nuovi mezzi e le nuove tencologie ci aiutando a realizzare software sempre più comodi e facili da utilizzare. Questo va certamente a vantaggio del nostro cliente/utilizzatore, ma va anche a nostro vantaggio, aprendo scenari di business fino a ieri preclusi.
In questi due articoli abbiamo visto come, in poco tempo, è possibile realizzare una applicazione abbastanza evoluta, comoda da usare e ricca di funzionalità come la generazione delle mappe. I mezzi, oggi, ci sono. Spazio alla fantasia dunque!

Michele Locuratolo

L'AUTORE

Michele Locuratolo è Software Architect per la *Mindbox S.r.l.* di Capurso (BA), società che si occupa di consulenza e sviluppo software. È cofondatore di *DotNetSide.org*, lo user group del Sud Italia il cui intento è quello di organizzare eventi di maggior spessore tecnico legati allo sviluppo con il .NET Framework. Il suo blog è raggiungibile all'indirizzo
http://www.dotnetside.org/blogs/mighell

FACCIAMO VIAGGIARE I DATI SU INTERNET

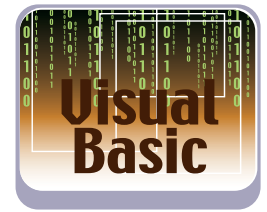
IN QUESTO ARTICOLO CREEREMO UN SERVER PERSONALIZZATO CHE ATTENDE RICHIESTE SU UNA PORTA PREDEFINITA. CONTEMPORANEAMENTE SVILUPPEREMO UN CLIENT PER LA CONNESSIONE. ET VOILÀ GETTATE LE BASI PER UNA CHAT, IL P2P O ALTRO....

È noto che la comunicazione tra computer avviene in base alle regole stabilite nei protocolli di trasmissione dati. I protocolli di trasmissione, in larghe linee, possono essere divisi in due tipi: protocolli di alto livello, che definiscono le regole per la trasmissione di singoli elementi come per esempio file o pacchetti, e protocolli di basso livello che si preoccupano di stabilire le regole per la trasmissione dei singoli byte o bit. Di ciò si è occupato l'ISO (International Standard Organization) già nel lontano 1979, quando definì il modello OSI (Open System Interconnection Model) come riferimento per la progettazione delle reti. Per fortuna, Visual Basic permette di gestire la comunicazione tra computer con i controlli Winsock ed Inet ed il programmatore non deve preoccuparsi di come i protocolli sono implementati. Il controllo Winsock consente di utilizzare i protocolli di trasmissione TCP/IP e UDP, mentre le applicazioni per il protocollo FTP possono essere gestite con il controllo Internet Transfer (Inet). Ricordiamo che il protocollo TCP (Transmission Control Protocol) è indicato per applicazioni Client/Server, cioè per le applicazioni che permettono lo scambio di dati tra una macchina denominata Server ed una o più macchine denominate Client, il protocollo UDP (User Datagram Protocol) è soprattutto indicato per applicazioni di tipo Peer to Peer (paritetiche), mentre il protocollo FTP (File Tansfer Protocol) è principalmente indicato per il trasferimento di file di grosse dimensioni, per esempio si usa quando si scaricano file da Internet, o quando dal proprio computer si trasferiscono file sul Server di un Internet Provider. In questo articolo presentiamo un'applicazione Client-Server, basata sul controllo Winsock e il protocollo TCP/IP, che consente di gestire la trasmissione di file tra computer. Prima, però, facciamo un breve cenno sui protocolli ed i controlli citati.

SCEGLIERE UN NOME PER IL COMPUTER

In generale per connettersi in remoto ad un computer bisogna conoscere il suo IP o il suo nome. Come è noto l'IP di un computer è una stringa numerica composta da 4 gruppi di 3 cifre separati da un punto. Ad esempio un IP può essere il seguente:125.1.2.124. Di solito l'IP di un computer non collegato in rete è 127.0.0.1 che rappresenta l'indirizzo dell'host locale a cui si può fare riferimento anche con il nome di localhost. Per conoscere il nome o l'IP di un computer collegato in rete si può procedere in vari modi, per esempio si può usare la scheda connessione alla Rete del pannello di controllo o il comando Dos Ipconfig.

TCP, UDP E FTP

Il protocollo TCP è nato nel mondo Unix ed ormai è disponibile su tutti i tipi di sistemi operativi. È un protocollo che garantisce la corretta trasmissione dei dati inviati e ricevuti, conservandone la corretta sequenza di trasmissione e ricezione. Insieme all'IP Protocol (Internet Protocol) che si occupa essenzialmente dell'instradamento dei pacchetti di dati, costituisce l'elemento fondante di Internet. TCP è un protocollo orientato alla connessione concepito per la trasmissione da punto a punto tra un computer Client e uno Server, per questo in un'applicazione che utilizza il protocollo TCP la cosa fondamentale da stabilire è se si tratta di un'applicazione Client o Server. In parole povere possiamo dire che il funzionamento del TCP è analogo al funzionamento del telefono tradizionale e cioè prima di dialogare bisogna stabilire una connessione. UDP è un protocollo utilizzato soprattutto per lo scambio di messaggi. L'UDP a differenza del TCP non richiede una connessione esplicita tra i computer cioè è di tipo connectionless. La mancanza di una connessione ne fa un protocollo poco affidabile ma veloce e semplice da utilizzare.

Con il protocollo UDP i computer possono avere un comportamento equivalente (possono trasmette e ricevere allo stesso modo, non è definita la figura del computer client e del computer server) per questo le applicazioni che utilizzano l'UDP si possono considerare di tipo Peer to Peer. Un'altra caratteristica di questo protocollo è che può essere usato per inviare messaggi in broadcast, cioè contemporaneamente su più nodi della rete. L'FTP (File Tansfer Protocol) è un protocollo che nasce nel 1972 nel MIT (Massachussets Institute of Technology), le specifiche dell'FTP vengono descritte nella RFC 114. L'FTP unisce i comandi del TCP/IP e del Telnet Protocol (Protocolli sviluppati con ARPAnet), si basa sul codice ASCII ed in particolare ne usa, solo, la parte inferiore (ASCII a 7 bit). I comandi dell'FTP oltre a controllare il flusso dei file, da e verso il Server, consentono, quando si trasmettono file, di interagire con il processo in atto sul Server. Ai Server FTP è possibile accedere anche attraverso un Web Browser, in questo caso l'indirizzo da specificare è del tipo ftp://ftp.microsoft.com.

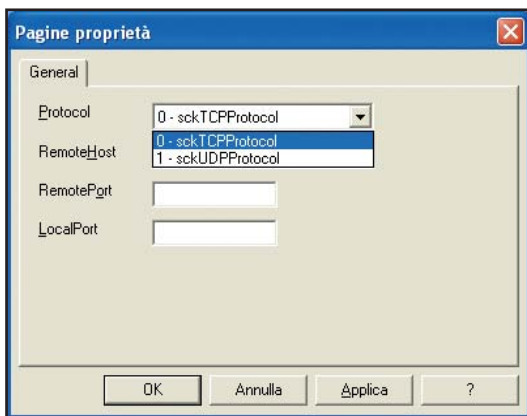
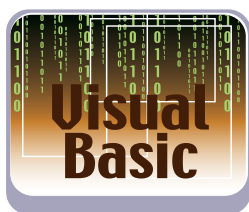



Fig. 1: **La finestra delle proprietà del Winsock**

I CONTROLLI WINSOCK E INTERNET TRANSFER

Come accennato Winsock è il controllo che consente di utilizzare i protocolli di trasmissione UDP e TCP, esso fa parte della libreria *Microsoft Winsock Control 6.0* (in cui è presente l'OCX MSWINSCK); di seguito, brevemente, ne descriviamo le proprietà, i metodi e gli eventi più importanti. Iniziamo dalla proprietà che consente di scegliere il tipo di protocollo, cioè *Protocol* che è disponibile in fase di progettazione, nella finestra delle proprietà, e a run-time, i valori che supporta sono: *sckTCPProtocol* e *sckUDPProtocol*. La proprietà *LocalHostName*, invece, restituisce il nome della macchina su cui si esegue l'applicazione. È una proprietà di sola lettura, per esempio nel modo seguente si può conoscere il nome del proprio computer:

```
Msgbox (Winsock1. LocalHostName)
```

La proprietà *LocalPort* imposta o restituisce il numero della porta locale. Per il Client è la porta locale dalla quale sono inviati i dati, per il Server, invece, è la porta locale di attesa da dove riceveranno i dati. Le porte UDP e TCP disponibili per le applicazioni sono 65536, la porta che di solito viene utilizzata con TCP/IP è la 1023. Questa proprietà è settabile in fase di esecuzione e di progettazione. *RemoteHost* restituisce o imposta l'identificativo dell'host remoto, è una proprietà disponibile sia in fase di progettazione che d'esecuzione. Questa può essere impostata come indirizzo IP o come DNS. *RemotePort* restituisce o imposta la porta dell'host remoto; dopo che è impostata la proprietà Protocol la *RemotePort* è impostata sulla porta predefinita e cioè la 80 per il protocollo HTTP. La proprietà *State* restituisce lo stato del controllo, è di sola lettura e non è disponibile durante la fase di progettazione.
Ora vediamo i principali eventi: Close si verifica quando il computer remoto chiude la connessione. ConnectionRequest viene intercettato sul Server quando il Client richiede una connessione. DataArrival è l'evento che avviene, quando arrivano nuovi dati al Server o al Client. Error si verifica, quando, durante una connessione, c'è un errore. Infine c'è l'evento SendProgress (con parametri BytesSent e BytesRemaining), generato durante l'invio dei dati, che può essere usato per avere informazioni sui byte inviati (Sent) e da inviare (Remaining).
I Metodi che utilizzeremo invece sono i seguenti: *Accept* viene utilizzato nell'evento *ConnectionRequest* per accettare una connessione, ad esso viene passato un parametro nominato *RequestID* che può servire per identificare il Client. Connect instaura una connessione con il computer remoto, richiede come parametro l'indirizzo della macchina a cui ci si vuole connettere e il nome della porta utilizzata dal computer per la connessione. *Close* termina una connessione TCP sia da applicazioni Client sia Server. *GetData* estrae un blocco di dati dal buffer e lo memorizza in una variabile di tipo *Variant*. *Listen* pone il Server in ascolto delle richieste dei Client. *SendData* invia i dati ad un computer

remoto; facciamo notare che quando si devono inviare dati binari bisogna utilizzare una matrice di Byte.
Il controllo Internet Transfer (Inet) consente di attivare una connessione FTP o HTTP e di eseguire dei comandi. I tipi di comandi che possono essere eseguiti dipendono dal protocollo di trasmissione selezionato cioè HTTP oppure FTP. I principali metodi del controllo sono: *OpenURL* ed Execute. OpenURL apre un elemento (pagina, directory) che si trova all'URL specificato. La sintassi è la seguente:

```
object.OpenUrl URL [,datatype]
```

URL è l'indirizzo dove si trova l'elemento che deve essere aperto. *DataType* è un parametro opzionale che specifica il tipo di dato da leggere: 0 dato stringa, 1 dato byte array.
Execute permette di inviare un comando ad un Server. La sintassi è la seguente:

```
object.Execute url, operation, data,
                        requestHeaders
```

URL indica l'URL da utilizzare, se non specificato verrà utilizzato quello di *OpenUrl*. *Operation* è una stringa che specifica il tipo di operazione da eseguire. Data è una stringa che specifica le informazioni che verranno utilizzate dall'operazione. RequestHeaders è una stringa che specifica informazioni supplementari. Le proprietà principali del controllo sono: *AccessType, Protocol, URL*. Per esempio *AccessType* è utilizzata per specificare il tipo di accesso alla rete Internet, infatti come è noto la connessione di un computer client ad Internet può avvenire direttamente o tramite Proxy che eventualmente funge da filtro, per questo i valori che la proprietà può assumere sono: cNamedProxy, icDirect e IcUseDefault. Se su un form inserite un controllo Inet e un RichTextBox con il seguente esempio potete aprire un file presente in uno spazio ftp.

```
RichTextBox1.Text = Inet1.OpenURL _
(InputBox("URL", ,
 "FTP://ftp.sitoftp.it/miofile.zip"))
```

Per eseguire il codice precedente dovete sostituire sitoftp con il sito dal quale volete scaricare il file. Naturalmente, per eseguire

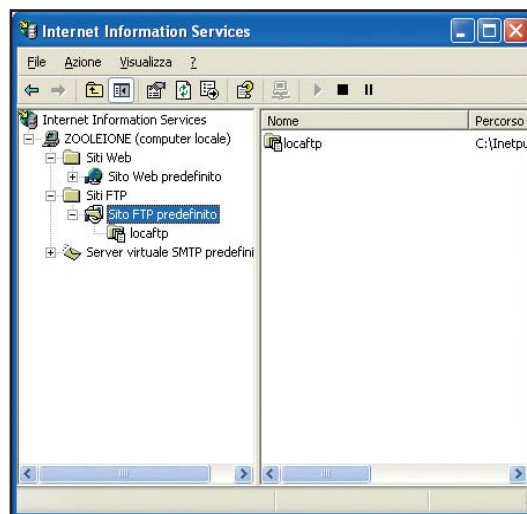



Fig. 2: *Il server ftp di IIS*

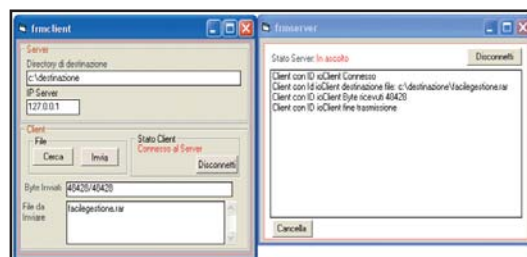



Fig. 3: *Le form dei progetti Client e Server*

l'operazione, dovete avere gli opportuni permessi di accesso.

APPLICAZIONE CLIENT-SERVER

L'applicazione basata sul Winsock è composta da due progetti: uno Client ed uno Server. In particolare l'applicazione Client permette di ricercare dei file in locale ed inviarli al Server. A tal fine il Client deve conoscere l'IP del Server e la porta che il Server utilizza per ricevere le richieste. Oltre a ciò, per semplificare, ipotizziamo che il Client conosca la directory del Server dove vengono trasferiti i file. Per sincronizzare i messaggi, tra il Server ed il Client, abbiamo definito il semplice protocollo schematizzato nella Tabella 1. Notate che i messaggi di sincronizzazione sono di 7 caratteri e terminano con due punti. Per

I COMANDI IPCONFIG E PING

Sicuramente i comandi più utilizzati dai gestori delle reti aziendali sono Ipconfig e Ping. Ipconfig permette di conoscere il nome e l'IP di un computer connesso in rete. Ping, invece, permette di capire se un computer è raggiungibile da una certa postazione e con quale velocità si comunica con esso. La sintassi è la seguente Ping Nome dove Nome può essere l'IP o il nome di un computer. Per utilizzare questi comandi basta aprire una sessione Prompt dei Comandi (MsDos) e al Prompt scrivere il comando e dare l'invio.

Messaggi dal Client al Server	
MSG-CLN:	Invio il Nome del Client
MSG-DES:	Invio il Nome del File
MSG-EOF:	Ho finito d'inviare il File
Messaggi dal Server al Client	
MSG-OKC:	Ok sei Connesso
MSG-OKI:	Ok puoi iniziare ad inviare i blocchi di byte
MSG-RIC:	Blocco Byte Ricevuto
MSG-CLC:	Ti ho sconnesso

Tabella 1 Il nostro Protocollo di trasmissione

esempio quando il Client deve inviare il suo nome invierà MSG-CLN:nomeclient, mentre se deve comunicare che ha finito d'inviare i blocchi di dati invierà MSG-EOF:
Presentiamo tutto il codice del progetto Server e soltanto la parte principale del codice del progetto Client

1 Create due progetti EXE e nominateli rispettivamente ioClient e ioServer. In entrambi i progetti referenziate l'oggetto Winsock. Nell'ioClient referenziate anche la libreria del CommonDialog.

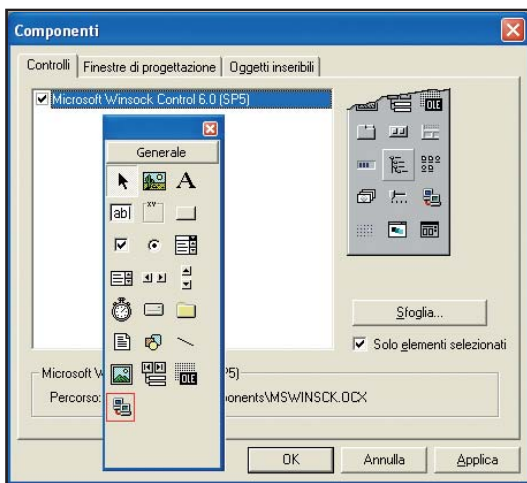



Fig. 4: ***Il componente Winsock***

2 Sulla Form di ioClient oltre al controllo Winsock (nominato WskClient) predisponete i seguenti elementi: due textbox per specificare una directory e l'IP del Server; un controllo CommonDialog, due pulsanti (nominati cerca ed invia) e un textbox (nominato txtfile). Questi elementi sono utilizzati per selezionare ed inviare i file al Server. Infine prevedete un altro pulsante (nominato connetti) e due label (nominate labelstato e labelbyte) che utilizzeremo per connetterci al Server e per mostrare lo stato della connessione e il numero di byte inviati. Disponete il tutto come nella figura 5.

Fig. 5: ***La Form del Client in fase di progettazione***

3 Sulla Form di ioServer oltre al controllo Winsock (nominato WskServer), predisponete i seguenti elementi: una listbox per mostrare gli eventi della trasmissione, cioè quale client si connette, che file invia ecc; due pulsanti nominati Disconnetti e Cancella che rispettivamente permettono di disconnettere il Client connesso e di cancellare la lista degli eventi. Inoltre anche in questo caso prevedete una label per visualizzare lo stato del Server. Disponete il tutto come nella figura 6.

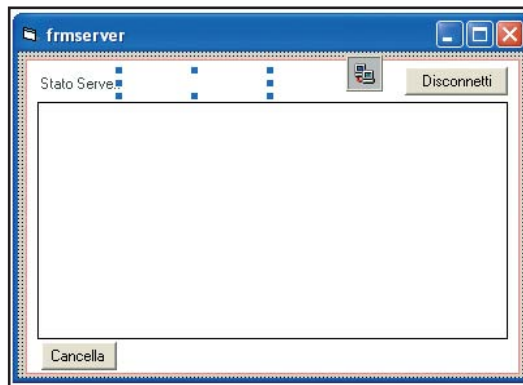
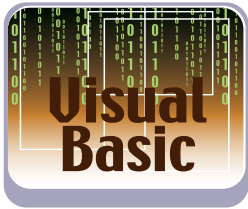



Fig. 6: ***La Form del Server in fase di progettazione***

COSA È UN WEB SERVICE ?

I web service sono i nuovi protagonisti dell'informatica distribuita. I web service possono essere utilizzati per vari scopi, da semplici contenitori/fornitori di informazioni di utilità generale (previsioni meteorologiche, annunci di vario tipo, ecc), a veri e propri servizi stand-alone o integrabili in altre applicazioni. Alle funzionalità offerte dai web service si può accedere con dei client che utilizzano il protocollo SOAP (Simple Object Access Protocol). Il protocollo SOAP è concepito per standardizzare lo scambio di messaggi tra computer e web service, esso non è legato ad un particolare linguaggio di programmazione o sistema operativo. Gli elementi fondanti del SOAP sono il Remote Procedure Calls (RPC), il protocollo HTTP e l'XML.

4 Iniziamo a presentare il codice del progetto ioServer. In particolare vediamo le dichiarazioni e la Form_Load.

```
Dim FileDest As String
Dim ByteRivevuti As Long
Dim numfile As Integer
Dim nomeclient As String
Private Sub Form_Load()
 WskServer.Protocol = sckTCPProtocol
 WskServer.LocalPort = 1023
 WskServer.Listen
 labelstate = "In ascolto"
 disconnetti.Visible = False
End Sub
```

Con il codice precedente oltre a mettere il Server in Listen, s'imposta il protocollo e la porta da cui il Server riceverà i dati. Nel nostro caso impostiamo una LocalPort dato che il Server e il Client si trovano sullo stesso Computer.

5 Il Server per gestire le richieste del Client utilizza gli eventi WskServer_Connection Request e WskServer_DataArrival. In particolare con il primo accetta la richiesta di connessione del Client ed invia il messaggio MSG-OKC: descritto in tabella 1. Nell'altro evento, invece, elabora i messaggi inviati dal Client e predispone le azioni per soddisfare le sue richieste. Tra l'altro invia i messaggi MSG-OKI: e MSG-RIC:.

```
Private Sub WskServer_ConnectionRequest( _
ByVal requestID As Long)
 If WskServer.State <> sckClosed Then
                                   WskServer.Close
  WskServer.LocalPort = 1023
  WskServer.Protocol = sckTCPProtocol
  WskServer.Accept requestID
  WskServer.SendData "MSG-OKC:"
End Sub
```

Facciamo notare che le istruzioni in ConnectionRequest non sono ridondanti ma necessarie, dato che nell'Help di VB c'è scritto "è consigliabile utilizzare il metodo Accept in una nuova istanza del controllo anziché nel controllo in attesa"

```
Private Sub WskServer_DataArrival
                              (ByVal bytesTotalAs Long)
 Dim Buffer As String
 Dim Risp As Integer
 On Error GoTo Errore
 WskServer.GetData Buffer
 Select Case Left(Buffer, 8)
  Case "MSG-CLN:" 'nome client
   nomeclient = Right(Buffer, Len(Buffer) - 8)
   List1.AddItem "Client con ID " + nomeclient + "
                                         Connesso"
   disconnetti.Visible = True
  Case "MSG-EOF:" 'fine del file
   List1.AddItem "Client con ID " + nomeclient _
   & " Byte ricevuti " & ByteRivevuti
   Close #numfile
   List1.AddItem "Client con ID " _
   & nomeclient + " fine trasmissione"
  Case "MSG-DES:" 'nome file destinazione
   FileDest = Right(Buffer, Len(Buffer) - 8)
   numfile = FreeFile
   On Error Resume Next
   If Len(Dir(FileDest)) > 0 Then
    Risp = MsgBox("Il file esiste sostituisco?", _
    vbCritical + vbQuestion + vbYesNo, "Server")
    If Risp = vbYes Then
     Kill FileDest
    Else
     'inviare errore al Client
     Unload Me
    End If
   End If
   Open FileDest For Binary As #numfile
    List1.AddItem "Client con Id " _
    & nomeclient + " destinazione file: " + FileDest
    WskServer.SendData "MSG-OKI:"
  Case Else
   ByteRivevuti = ByteRivevuti + Len(Buffer)
   Put #numfile, , Buffer
   WskServer.SendData "MSG-RIC:"
  End Select
  Exit Sub
Errore:
  MsgBox Err.Description, vbCritical, "Server"
  Unload Me
End Sub
```

6 Il Server per disconnettere il Client e pulire la ListBox utilizza il seguente codice.

```
Private Sub disconnetti_Click()
 labelstate = "In ascolto"
 disconnetti.Visible = False
 List1.Clear
 WskServer.SendData "MSG-CLC:"
```

CHE COSA VUOL DIRE LA SIGLA SSL?

SSL è il più comune protocollo (client/server) di sicurezza adoperato sulla rete. SSL per proteggere i dati, utilizza una combinazione di certificati digitali (chiave pubbliche/private) e crittografia. I certificati ammessi possono essere sviluppati secondo vari standard, tra cui lo standard X.509. Naturalmente anche Internet Explorer utilizza SSL, controllate la scheda Contenuto della maschera Opzioni Internet.

```
 WskServer_Close
End Sub
Private Sub WskServer_Close()
 List1.AddItem "Client con ID " _
 & nomeclient + " Connessione chiusa"
 WskServer.Close
 Form_Load
End Sub
Private Sub Cancella_Click()
 List1.Clear
End Sub
```

Nella Disconnetti notate l'invio del messaggio MSG-CLC:

7 Ora passiamo al Client. Di seguito presentiamo le parti principali del progetto. In particolare la parte di codice che gestisce la richiesta di una connessione è composta dalla connetti_Click e dalla ConnettiServer riportate di seguito.

```
Private Sub connetti_Click()
  If connetti.Caption = "Connetti" Then
   ConnettiServer
  Else
   WskCliente.Close
   Form_Load
  End If
End Sub

Private Sub ConnettiServer()
 On Error GoTo errore
  Dim Contr As Long
  If WskCliente.State <> sckClosed Then
                              WskCliente.Close
  WskCliente.Connect Me.txtipserver, 1023
  Contr = 0
  While (Not HaRisposto) And (Contr < 100000)
   DoEvents
   Contr = Contr + 1
  Wend
  If Contr >= 100000 Then
   MsgBox "Non è possibile connettersi al
                      Server",vbCritical, "Client"
   WskCliente.Close
   Exit Sub
  End If
  WskCliente.SendData "MSG-CLN:" +
                                  App.EXEName
  Me.connetti.Caption = "Disconnetti"
  Exit Sub
errore:
  MsgBox Err.Description
End Sub
```

Notate che la Caption del pulsante Connetti, inizialmente impostata su Connetti, è cambiata in Disconnetti dopo che il Server ha accettato la connessione. Inoltre notate che nella ConnettiServer, il Client capisce che il Server ha accettato la sua richiesta controllando il valore del Flag HaRisposto che è impostato nella WskCliente_DataArrival descritta nel punto successivo. Il Client dopo aver constatato ciò invia il suo nome con il messaggio MSG-CLN:.

8 Il Client (come nel caso del Server) utilizza l'evento WskCliente_DataArrival per elaborare i messaggi inviati dal Server, di seguito il codice.

```
Private Sub WskCliente_DataArrival _
(ByVal bytesTotal As Long)
 Dim recBuffer As String
 WskCliente.GetData recBuffer
 Select Case Left(recBuffer, 8)
  Case "MSG-RIC:"  'Blocco ricevuto
   Ricevuto = True
  Case "MSG-OKC:"  'Ok connesso
   HaRisposto = True
  Case "MSG-OKI:"  'Ok puoi iniziare ad inviare
   InviaFile
  Case "MSG-CLC:" 'connessione closed
   WskCliente_Close
   Unload Me
  Case "Msg-Err:"  'errore
    'non gestito
 End Select
End Sub
```

CONCLUSIONE

Abbiamo introdotto le principali tecniche per la creazione di applicazioni Client-Server per Internet. Nello spazio a disposizione però non abbiamo potuto illustrare tutto il codice del progetto Client che trovate nel CD allegato alla rivista.

Massimo Autiero

FILE TRANSFER PROTOCOL IN BREVE

In una configurazione Client/Server FTP, il Server è un sistema in cui sono archiviati dei file che possono essere manipolati, in remoto, con un FTP Client. Sul Server FTP sono definiti gli Account dei Client. I Client FTP dopo aver specificato Username e Password possono accedere, in lettura/scrittura, ad una parte oppure a tutte (dipende dal tipo di Account) le directory del Server. Su alcuni Server, di solito, è definito l'account anonymous (con password un indirizzo di posta elettronica o guest) che contiene le directory pubbliche del Sito.

COSTRUIRE UNO SKYPEBOT

SE VI PIACE SKYPE, PREPARATEVI A DIVENTARNE AMICI INTIMI. IN QUESTO ARTICOLO SCRIVEREMO UN PROGRAMMA CHE RISPONDE AI MESSAGGI DI CHAT USANDO LE API PUBBLICHE DI SKYPE E IL LINGUAGGIO RUBY.

Siete pronti a fare una scorpacciata di programmazione? Questo articolo mette parecchia carne al fuoco: Skype, il linguaggio Ruby, gli oggetti COM, i Web Service... Alla fine vi servirà del bicarbonato, ma sono certo che vi divertirete molto. Staccate il telefono, chiudete a chiave la porta e cominciamo!

L'APPLICAZIONE CHE FA PARLARE LA GENTE

Cominciamo da Skype. Esistono delle API pubbliche per estendere il client di Skype (https://developer.skype.com/DevZone), ma purtroppo non per accedere direttamente al server. Per fortuna le API del client sono sufficienti per scrivere programmi interessanti come gli SkypeBot. Uno SkypeBot è un programma che fingere di essere un utente di Skype. Di solito intercetta i messaggi in arrivo e risponde a tono. In questo articolo scriveremo uno SkypeBot che cerca di rendersi utile fornendo un semplice servizio di conversione.

Le API standard di Skype richiedono il linguaggio C, ma esistono valide alternative. Esiste un wrapper Java delle API, e anche un wrapper COM che espone il client di Skype come un oggetto Windows. Noi useremo quest'ultimo (Skype4COM) per accedere alle API con un linguaggio script. Dopo aver installato Skype, usate skype_contacts.exe, che troverete sul CD per installare i controlli ActiveX. La documentazione dice di registrare il server COM, ma in realtà l'installer dovrebbe pensarci da solo. Se qualcosa va storto e volete registrare manualmente la DLL, andate nella directory Skype\ActiveX\Contacts e date il comando:

```
regsvr32 /u skype4com.dll
```

Ora Skype è disponibile come un oggetto COM. Ci serve solo un buon linguaggio per parlargli.

UN GIOIELLINO DI LINGUAGGIO

Ruby è un linguaggio script completamente object-oriented, semplice, produttivo e divertente. Per leggere questo articolo dovete avere delle buone basi di programmazione, ma non è necessario che conosciate già Ruby o la programmazione object-oriented. Non vi sarà difficile seguire gli esempi, soprattutto se avete già usato PHP o Perl. Installate RubyInstaller per Windows, che troverete sul CD. L'installer contiene il linguaggio e una serie di utility, incluso FreeRIDE (un IDE per Ruby) e SciTE (un semplice editor con sintassi colorata).

Durante i primi esperimenti vi conviene tenere aperto l'interprete irb, che permette di eseguire dei comandi e vedere immediatamente il risultato. Se scoprite di non riuscire a scrivere alcuni caratteri in irb (a me succede con le parentesi quadre della mia tastiera italiana), lanciatelo con lo switch noreadline:

```
irb --noreadline
```

Potete usare qualsiasi editor per scrivere i file Ruby, che sono file di testo con estensione .rb. Se volete un editor fatto apposta, usate SciTE. SciTE lancia lo script quando si preme F5. Se preferite lanciare uno script dalla riga di comando, scrivete:

```
ruby <nome_del_file>
```

È arrivato il momento di sporcarsi le mani con un po' di codice

MICROCORSO: RUBY E WINDOWS OLE

Siete pronti per un corso ultrarapido di programmazione Windows in Ruby? Impareremo come mandare messaggi ad un oggetto COM da Ruby e come ricevere eventi dall'oggetto. Durata totale del corso: 15 minuti abbondanti.

Come oggetto COM useremo il buon vecchio Internet Explorer. Scrivete e lanciate questo script Ruby:

```
require 'win32ole'
ie =WIN32OLE.new('InternetExplorer.Application')
ie.visible = true
ie.navigate('http://www.google.co.jp')
```

Internet Explorer dovrebbe aprirsi e accedere alla pagina giapponese di Google. Vediamo riga per riga come funziona lo script:

```
require 'win32ole'
```

L'istruzione require esegue un file di nome win32ole.rb, che fa parte delle librerie del linguaggio. Non solo il codice viene eseguito, ma tutte le variabili e le funzioni ivi definite diventano visibili dallo script (da mesi sognavo di usare la parola "ivi" in un articolo). Il file definisce tra l'altro una classe che si chiama WIN32OLE e trasforma oggetti Windows in oggetti Ruby. Ora possiamo usare il metodo new per vedere l'Internet Explorer come un oggetto Ruby:

```
ie = WIN32OLE.new('InternetExplorer.Application')
```

Il parametro del new è una stringa che contiene l'identificativo dell'oggetto COM. Una stringa in Ruby può essere delimitata sia da singoli apici che da virgolette.
Se non siete abituati alla programmazione a oggetti, tutto questo parlare di metodi e classi potrebbe confondervi. L'unica cosa che dovete sapere per seguire questi esempi è che un oggetto si crea con il metodo new e possiede delle proprietà e dei metodi. Le proprietà sono simili ai campi di una struct in C. Ad esempio, l'oggetto ie ha una proprietà booleana visible. Se visible è true, allora la finestra di Internet Explorer è visibile:

```
ie.visible = true
```

I metodi sono dei comandi che possiamo chiamare in modo simile alle funzioni. Ad esempio, Internet Explorer ha un metodo navigate che apre una URL:

```
ie.navigate('http://www.google.co.jp')
```

Come vedete, parlare agli oggetti COM con Ruby è facile. Ma a volte vorremmo che fossero loro a parlare a noi. Per fare questo possiamo metterci in ascolto per ricevere eventi dall'oggetto. Per ricevere un evento quando qualcuno apre una URL in Internet Explorer, aggiungete queste righe allo script:

```
ev = WIN32OLE_EVENT.new(ie,
                        'DWebBrowserEvents')
ev.on_event("NavigateComplete") {|url|
  print url, "\n"
}
loop { WIN32OLE_EVENT.message_loop }
```

La prima riga usa la classe WIN32OLE_EVENT delle librerie win32ole per creare un oggetto che riceve gli eventi del browser. *DWebBrowser Events* è una delle interfacce messe a disposizione da Internet Explorer per gestire gli eventi (per scoprirlo abbiamo dovuto consultare la complicatissima documentazione di Internet Explorer sul sito di Microsoft). Uno degli eventi, *NavigateComplete*, segnala che Explorer ha aperto una nuova pagina. Quando riceviamo l'evento, vogliamo stampare la URL della pagina sullo schermo. Il codice per ottenere questo risultato è un po' complicato, e merita uno sguardo attento.
Il metodo *on_event* prende come parametro il nome dell'evento che vogliamo ricevere. Quello che segue, delimitato da parentesi graffe, è un costrutto Ruby che non esiste nella maggior parte degli altri linguaggi: un blocco. Non abbiamo spazio in questo articolo per imparare bene cosa sono e come funzionano, ma in questo caso possiamo semplicemente pensare al blocco come ad un "callback". Questo codice dice: "quando arriva un evento *NavigateComplete*, chiama questo blocco passandogli come parametro la URL appena aperta". Il codice del blocco stampa la URL sullo schermo, seguita da un carattere di a-capo.
Ora dobbiamo solo attendere che Internet Explorer ci mandi i suoi eventi. Ci serve un ciclo infinito, altrimenti il programma termina subito anziché restare in ascolto. Ma se vogliamo "consumare" gli eventi in arrivo, il ciclo infinito non basta. Dobbiamo anche eseguire continuamente il "message loop" di Windows:

```
loop { WIN32OLE_EVENT.message_loop }
```

ALTA VELOCITÀ

Il linguaggio Ruby (http://www.ruby-lang.org) è in circolazione da qualche anno, ma solo recentemente è diventato una star. Merito soprattutto di Ruby On Rails (http://www.rubyonrails.org), un framework per la programmazione Web talmente semplice e produttivo da sembrare magico. Parleremo in dettaglio di "Ruby su Rotaie" nei prossimi numeri di IoProgrammo.

La parola loop dice a Ruby di eseguire per sempre il codice tra parentesi graffe, finché non interromperete l'esecuzione del programma. Provate a lanciare questo script e visitate un po' di pagine in Internet Explorer.
I nomi delle URL dovrebbero apparire sullo schermo. Se tutto funziona, ne sapete abbastanza per affrontare le API di Skype.

IL PAPPAGALLO CAPOVOLTO

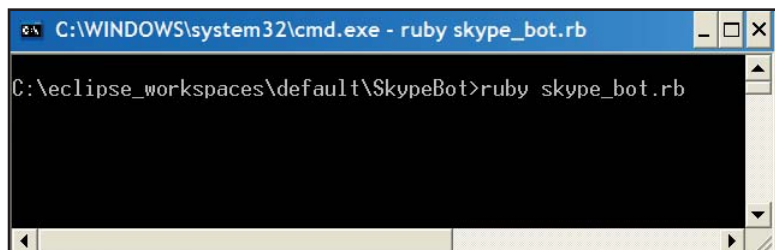
E' il momento di scrivere un semplice SkypeBot. Per seguire le chiamate alle API Skype4COM, potete tenere aperta la documentazione online all'indirizzo *https://developer.skype.com/Docs /Skype4COMLib.*

```
require 'win32ole'
skype = WIN32OLE.new("Skype4COM.Skype")
skype.Client.Start if !skype.Client.IsRunning
skype.Attach
MESSAGE_STATUS_RECEIVED =
        skype.Convert.TextToChatMessageStatus
                                  ("RECEIVED")
MESSAGE_TYPE_SAID =
    skype.Convert.TextToChatMessageType("SAID")
ev = WIN32OLE_EVENT.new(skype)
ev.on_event('MessageStatus') {|msg, status|
  if status == MESSAGE_STATUS_RECEIVED &&
             msg.Type == MESSAGE_TYPE_SAID
    skype.SendMessage(msg.FromHandle,
                          msg.Body.reverse)
  end
}
loop { WIN32OLE_EVENT.message_loop }
```

ALZATI, MIA CREATURA!

Lo SkypeBot intercetterà i messaggi riservati all'utente che ha fatto login sul client di Skype. Purtroppo è possibile eseguire un solo client su ciascun computer, quindi servono due computer (o l'aiuto di un amico) per provare comodamente lo SkypeBot. Ecco come fare:

1 - Lanciate Skype. Potete usare il vostro utente ufficiale, o creare un nuovo utente apposta per lo SkypeBot. Per disconnettere l'utente esistente e crearne uno nuovo, usate la voce Chiudi Collegamento del menu File, e poi cliccate su "Non hai un nome utente Skype?". Occhio a non dimenticare la password del vostro vero utente!

2 – Lanciate lo SkypeBot.

3 – Da un altro computer, usate Skype per mandare messaggi al bot. Tutti i messaggi in ingresso e in uscita saranno mostrati dal client di Skype nella solita finestra di chat. In alternativa potete provare il bot dallo stesso computer cambiando continuamente l'utente, e lanciando e fermando lo SkypeBot.

Esaminiamo il bot riga per riga:

```
require 'win32ole'
skype = WIN32OLE.new("Skype4COM.Skype")
```

Abbiamo usato win32ole per istanziare un oggetto che permette di mandare messaggi a Skype. Poi ci assicuriamo che il client di Skype sia aperto:

```
skype.Client.Start if !skype.Client.IsRunning
```

Questa riga si legge "lancia il client di Skype se non sta già girando".
Start e IsRunning sono entrambi metodi dell'oggetto Client delle API. In Ruby, a differenza di molti altri linguaggi, le parentesi tonde nella chiamata di un metodo sono opzionali, e in questo caso le abbiamo omesse. Avremmo anche potuto scrivere:

```
skype.Client.Start() if !skype.Client.IsRunning()
```

L'if si può anche mettere prima della condizione come nella maggior parte dei linguaggi, ma in questo caso si deve usare end per chiudere il blocco di istruzioni:

```
if !skype.Client.IsRunning
  skype.Client.Start
end
```

Ora che sia, certi che il client di Skype stia girando, attacchiamoci a lui come simpatici parassiti:

```
skype.Attach
```

Le righe successive servono a definire due costanti che torneranno utili tra poco:

```
MESSAGE_STATUS_RECEIVED =
        Skype.Convert.TextToChatMessageStatus
                                  ("RECEIVED")
```

```
MESSAGE_TYPE_SAID =
        skype.Convert.TextToChatMessageType("SAID")
```

Per definire una costante in Ruby basta usare un nome che inizia con una lettera maiuscola. Per convenzione si usano solo lettere maiuscole, e si separano le parole con degli underscore.
Lo stato di un messaggio è “RECEIVED” per i messaggi ricevuti, e il tipo di un messaggio è “SAID” per i normali messaggi di chat, “SETTOPIC” per i messaggi che cambiano il titolo della chat (quelli inviati attraverso il comando skype /topic), eccetera.
Noi siamo interessati ai messaggi di chat in ingresso, e abbiamo usato l'oggetto *Convert* di Skype per convertire le stringhe identificative di questi stati nelle relative costanti. Passiamo agli eventi:

```
ev = WIN32OLE_EVENT.new(skype)
ev.on_event('MessageStatus') {|msg, status|
  if status == MESSAGE_STATUS_RECEIVED &&
                     msg.Type == MESSAGE_TYPE_SAID
    skype.SendMessage(msg.FromHandle,
                                  msg.Body.reverse)
  end
}
loop { WIN32OLE_EVENT.message_loop }
```

La gestione degli eventi e il loop infinito sono simili a quelli dell'esempio nel paragrafo precedente. Questa volta restiamo in attesa dell'evento *MessageStatus*, generato da Skype ogni volta che succede qualcosa che riguarda un messaggio. L'evento ha due argomenti: un messaggio (un oggetto *ChatMessage* di Skype) e uno status. Quando riceviamo l'evento, analizziamo il messaggio e lo status: se si tratta di un messaggio di chat in arrivo, allora rispondiamo con il metodo *SendMessage* di Skype.
SendMessage prende come argomenti il destinatario del messaggio (che è il mittente del messaggio precedente, ottenuto con il metodo *FromHandle*) e il corpo del messaggio (che è lo stesso del messaggio precedente, ottenuto con il metodo *Body*). Per rendere le cose più interessanti abbiamo invertito il corpo del messaggio, che è una normale stringa, con il metodo *reverse* delle stringhe di Ruby.
Il box Alzati, mia creatura! spiega come lanciare e usare lo SkypeBot. Se provate a mandare un messaggio al bot:

```
Salve!
```

lui risponderà con lo stesso messaggio invertito:

```
!evlaS
```

MICROCORSO: RUBY E I WEB SERVICE

Non siete stanchi, vero? E' il momento di un nuovo microcorso. Titolo: “Come connettersi ad un Web Service da Ruby”. Durata: 10 minuti. Se sapete già cos'è un WSDL, potete saltare il prossimo paragrafo.
Un Web Service (che d'ora in poi chiameremo WS) è un servizio accessibile da un programma tramite HTTP. Esempi classici sono un validatore di carte di credito, o un report dell'andamento delle borse. Un WS pubblica la propria interfaccia con un linguaggio basato su XML che si chiama WSDL (Web Service Description Language). Il WSDL può specificare ad esempio che il WS fornisce una procedura chiamata *weather_report*, che restituisce le previsioni del tempo per una certa città. Se passo al WS il nome della città, lui mi risponde con le previsioni. Ma per scambiare questi dati tra client e WS li devo codificare con qualche standard. Di solito si usa SOAP (Simple Object Access Protocol), un altro standard basato su XML.
Per prima cosa dobbiamo trovare un WS che ci piace. Ne useremo uno che fa le conversioni di valuta in tempo reale. Il WSDL è all'indirizzo:

```
http://www.xmethods.net/sd/
                    CurrencyExchangeService.wsdl
```

Questo WSDL mostra che il Web Service accetta un messaggio chiamato *getRate*, che prende due codici di valuta e restituisce il tasso di cambio. Ecco un programmino Ruby che accede al servizio e stampa il tasso di conversione tra Euro e dollaro:

```
require 'soap/wsdlDriver'
driver_factory =
        SOAP::WSDLDriverFactory.new("http://www.
    xmethods.net/sd/CurrencyExchangeService.wsdl")
soap = driver_factory.create_driver
print "Euro/dollaro: ", soap.getRate("euro", "usd")
```

La prima riga importa nel programma la libreria standard di Ruby per l'accesso ai WS. Questa libreria definisce *WSDLDriverFactory*, un oggetto che è in grado di interpretare un WSDL. La seconda riga crea una nuova *WSDLDriverFactory*. La terza riga usa la factory per creare un oggetto che riceve tutti i messaggi definiti nel WSDL. Il risultato è un normale oggetto con i suoi metodi. Quando chiamate un metodo, in realtà state facendo una chiamata al WS. Riassumendo: prima abbiamo “avvolto” il WSDL in un oggetto che

sa come leggerlo e sa anche come costruisci attorno degli oggetti Ruby (una "fabbrica"); poi abbiamo usato questa fabbrica per creare un oggetto che riceve i nostri messaggi e li trasmette al WS.
Ora possiamo usare questo nuovo oggetto. Se ne occupa la quarta riga, che chiama il metodo *getRate* passando come parametri i codici di valuta per l'Euro e il dollaro USA. Il risultato è il tasso di cambio, che viene stampato sullo schermo (sempre che il Web Service sia disponibile). Mentre scrivo, un Euro vale 1,2735 dollari. Se non fosse per l'IVA, la dogana e le spese di spedizione, sarebbe il momento giusto per acquistare online dagli Stati Uniti.
E tutta la faccenda della codifica dei dati in SOAP? Ci pensa Ruby, voi non dovete preoccuparvene. Quindi il microcorso su come accedere ai Web Service in Ruby è finito. Mettiamo a frutto quello che abbiamo imparato.

UN ALTRO PO' DI RUBY

Ecco uno script Ruby che usa lo stesso Web Service del paragrafo precedente per convertire un importo di denaro da una valuta all'altra:

```
require 'soap/wsdlDriver'
currencies = SOAP::WSDLDriverFactory.
new("http://www.xmethods.net/sd/
                    CurrencyExchangeService.wsdl").
create_rpc_driver

def convert_money(msg)
  begin
    amount, from, ignored, to = msg.split(' ')
    rate = currencies.getRate(from, to)
    return [amount, from, "equivalgono a",
                    amount.to_i * rate, to].join(' ')
  rescue
    return "Esempio: 100 euro -> us"
  end
end
```

Il codice per accedere al WS è più o meno lo stesso di prima, e verrà eseguito la prima volta che lo script viene eseguito o caricato con require. Questa volta non abbiamo perso tempo a mettere la factory in una variabile temporanea: la creiamo, la usiamo per creare un oggetto currencies e poi la abbandoniamo al suo destino. Ruby permette di spezzare le righe con degli a-capo (con qualche limitazione), così evitiamo di offuscare il codice con righe troppo lunghe.
Poi lo script definisce una funzione convert_money che usa l'oggetto currencies. Una funzione in Ruby si definisce così:

```
def nome_della_funzione(argomento1,
                                    argomento2, ...)
  codice_della_funzione
end
```

La funzione convert_money riceve un messaggio che contiene l'importo e le valute, lo analizza rozzamente, lo usa per mandare i parametri al convertitore e restituisce il tasso di cambio. Osserviamo meglio le tre righe centrali della funzione:

```
amount, from, ignored, to = msg.split(' ')
```

Per prima cosa parliamo di quello che c'è a destra dell'assegnamento. Il metodo split spezza la stringa in un array di stringhe più piccole usando un carattere di separazione – in questo caso uno spazio. Abbiamo fatto l'ipotesi che il messaggio sia costituito da quattro "parole" separate da spazi, quindi il risultato dello split sarà un array di quattro stringhe. Questo array viene assegnato a quattro variabili. L'astuto Ruby farà la cosa giusta, e assegnerà a ciascuna variabile il corrispondente elemento dell'array: il primo elemento finisce in amount, il secondo in from, e così via. Se il messaggio non contiene esattamente quattro parole avremo un errore. Ne parleremo tra poco.
Ora abbiamo due variabili (*from e to*) che sono le due valute tra cui convertire, una (*amount*) che è l'importo di denaro, e una (*ignored*) che non serve a nulla, e possiamo appunto ignorare. La riga successiva usa il convertitore per ottenere il tasso di cambio:

```
rate = currencies.getRate(from, to)
```

Ora possiamo restituire una stringa che dà il risultato dell'operazione:

```
return [amount, from, "equivalgono a", amount.to_i
                                        * rate, to].join(' ')
```

Il risultato è una semplice stringa, che avremmo potuto costruire in modo più tradizionale – ma questa riga inutilmente virtuosistica ci dà la scusa per parlare un altro po' di Ruby. La roba tra parentesi quadre è un array definito al volo, che contiene i "pezzi" della stringa che vogliamo restituire al chiamante. L'elemento più importante è:

```
amount.to_i * rate
```

La variabile *amount* (che, lo ricordiamo, è una stringa) viene convertita in un intero con il metodo *to_i* e moltiplicata per il tasso di cam-

bio per ottenere l'importo finale.
Il metodo join degli array è il contrario del metodo split delle stringhe: concatena tutti gli elementi in una stringa, usando come separatore il carattere che gli passiamo. Quindi:

```
[10, "euro", "equivalgono a", 100, "pizze di fango del
                                   camerun"].join(' ')
```

ha come risultato:

```
10 euro equivalgono a 100 pizze di fango del
                                             camerun
```

I programmatori sono bravi ad immaginare modi in cui le cose possono andare storte, quindi sicuramente vi saranno venuti in mente molti scenari orribili.
La prima parola del messaggio potrebbe dare un errore quando cerchiamo di convertirla in numero; la seconda potrebbe non essere un codice di valuta, e causare un errore nella chiamata al Web Service; o magari il WS potrebbe essere non disponibile. In casi come questi avremo qualche tipo di errore.
La funzione *convert_money* gestisce gli errori in modo estremamente rozzo, con un blocco *begin-rescue-end*. Se si verifica un errore nel codice compreso tra *begin* e *rescue*, l'esecuzione del programma salta al codice compreso tra rescue e end. Se tutto va bene, questo codice non viene mai eseguito.

```
begin
        <codice che può generare errori>
rescue
        <codice da eseguire in caso di errore>
end
```

Nel nostro caso l'errore viene "gestito" rispondendo al chiamante con una stringa che spiega come comporre un messaggio valido. Questo modo allegro di gestire gli errori sarebbe inaccettabile in un programma in produzione, per parecchi motivi.
Se il messaggio è scritto bene ma il Web Service non funziona, l'utente riceve una risposta frustrante che lo colpevolizza senza ragione; la terza parola del messaggio viene ignorata, quindi può contenere qualsiasi schifezza; non abbiamo verificato che la libreria soap/wsdlDriver supporti la concorrenza, quindi non sappiamo se la funzione *process* può essere chiamata con sicurezza da più thread in parallelo; e così via dubitando. Ma dato che gestire bene gli errori richiede tempo e spazio, teniamoci questo semplice esempio così com'è. Se mai rivenderete questo programma, vi raccomando di scrivere una gestione degli errori più solida (e di presentarmi i vostri clienti). Salvate lo script in un file di nome *currency_converter.rb*. Come al solito lo si può importare con un require:

```
require 'currency_converter'
```

Quando lo importiamo, lo script viene eseguito e inizializza la variabile *currencies*. A questo punto possiamo chiamare la funzione *convert_money* (con un messaggio nel formato che abbiamo detto) per fare una conversione di valuta. Provate a lanciare *irb*, importare lo script e chiamare la funzione:

```
print convert_money("100 euro -> us")
```

Ora abbiamo tutto quello che ci serve per inserire un cambiavaluta nel nostro SkypeBot.

METTERE INSIEME I PEZZI

Bastano un paio di modifiche per trasformare l'inutile EchoBot in uno SkypeBot più dignitoso:

```
require 'win32ole'
require 'currency_converter'
<...>
ev.on_event('MessageStatus') {|msg, status|
  if status == message_status_received &&
                        msg.Type == message_type_said
    skype.SendMessage(msg.FromHandle,
                        convert_money(msg.Body))
  end
}
```

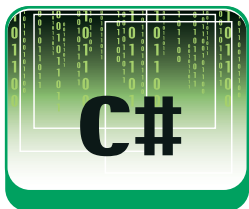
Niente male, come inizio. Ora si potrebbe aggiungere un comando per chiedere al bot le quotazioni di borsa. naturalmente dopo aver trovato un web service adeguato. Naturalmente si dovrebbe scrivere del codice per distinguere tra i vari comandi, in modo da poter usare sia il cambiavaluta che le quotazioni. E naturalmente non dovete limitarvi a bot che rispondono a semplici comandi di chat. Con un po' di fantasia e le API di Skype potreste ad esempio scrivere una "sveglia automatica" che vi fa una telefonata ad un'ora predefinita per ricordarvi che è ora di staccarvi dal PC. Sono sicuro che avete già idee migliore di queste, quindi vi lascio ai vostri esperimenti. Buon divertimento con le API di Skype!

Paolo Perrotta

SE IL WEB SERVICE VA GIÙ

I Web Service vanno e vengono. Il convertitore di valuta che abbiamo usato per i nostri esempi potrebbe non essere più disponibile quando leggerete questo articolo. Semplicemente cercate qualche altro WS semplice e modificate un po' il codice. Il WS ideale non dovrebbe richiedere autenticazione e dovrebbe avere almeno una chiamata che restituisce una semplice stringa. Potrete esplorare siti come http://www.webservicelist.com per trovare nuovi giocattoli.

UN PLUGIN PER GOOGLE DESKTOP

LI CHIAMANO GADGET! I PICCOLI PLUGIN CHE ESTENDONO LA PIÙ NOTA APPLICAZIONE PER LA RICERCA DI CONTENUTI "OFF-LINE". IMPARIAMO A COSTRUIRNE UNO CHE CI MOSTRA LA LISTA DEI "TASK" DA ESEGUIRE NEL CORSO DELLA GIORNATA

Sono ormai numerose le applicazioni che consentono l'indicizzazione del contenuto dei nostri hard disk al fine di garantire funzionalità di ricerca veloci ed immediate: Microsoft con Windows Desktop Search e Yahoo! Desktop Search sono tra i sistemi più diffusi. Ma quando si parla di motori di ricerca non può di certo mancare il motore di ricerca per eccellenza: Google. Con il suo Google Desktop Search il gigante di Mountain View entra a pieno titolo, e per la prima volta, nel mondo desktop. Google Desktop Search, oltre ad integrarsi pienamente con il motore di ricerca web e con la sua interfaccia, non si limita solo a fornire avanzate funzionalità di ricerca, ma fornisce anche una comoda sidebar costituita da un insieme di plugin, chiamati gadget, le cui funzionalità possono essere facilmente estese. In questo articolo impareremo a creare un plug-in per la sidebar di Google Desktop Search che consenta la rapida visualizzazione di una "task list" fornita da un nostro ipotetico applicativo. Purtroppo per motivi di spazio non sarà possibile mostrare tutto il codice necessario per il corretto funzionamento del plugin. Il codice completo è comunque disponibile nel CD allegato alla rivista.

CREAZIONE DELL'AMBIENTE

Come prima operazione dobbiamo impostare l'ambiente di sviluppo scaricando ed installando il Google Desktop SDK, il cui indirizzo è indicato nel **box laterale**. Il Google Desktop SDK installa esempi e documentazione per l'utilizzo delle API, oltre ai componenti COM che utilizzeremo nel nostro progetto. Apriamo Visual Studio .NET e creiamo un nuovo progetto Class Library in C#. Aggiungiamo un riferimento alle librerie *System.Windows.Forms* e alle librerie COM *Google Desktop Display API Type Library* e *Google Desktop Search API 1.1 Type Library*. Nel progetto utilizzeremo anche alcune interfacce e strutture per comunicare con le classi unmanaged e che possiamo trovare nel file imports.cs contenuto nel CD oltre che nelle librerie dell'SDK.

Tasks
id
title
description
date
done

Fig. 1: *La tabella che conterrà i nostri task*

CREAZIONE DEL PLUG IN

Aggiungiamo al nostro progetto un nuovo file TaskPlugin.cs. Inseriamo un riferimento al namespace *GoogleDesktopAPILib* e dichiariamo una classe *TaskPlugin* che implementa la classe base astratta *GoogleDesktopDisplayPluginHelper Class*, wrapper della classe *Google Desktop DisplayPluginHelper* contenuta nelle *Google Desktop Display API*, e le interfacce *IGoogleDesktopDisplayPluginHandler, IObjectWithSite e IPersistStreamInit*:

- *GoogleDesktopDisplayPluginHelperClass*: è la classe base astratta che fornisce tutte le funzionalità di base per la creazione di un display plugin o gadget;
- *IGoogleDesktopDisplayPluginHandler*: è l'interfaccia che fornisce i servizi di interazione con gli eventi della sidebar;
- *IObjectWithSite*: consente di impostare o recuperare il site, inteso come l'host del plugin, ed utilizzato per eventualmente gestire le proprietà del plugin;
- *IPersistStreamInit*: l'interfaccia consente di

rendere persistente lo stato del controllo e di ricaricarlo al successivo avvio della sidebar;

Otteniamo quindi un codice simile a questo:

```
[GuidAttribute("766bcd9b-0b64-469c-b396
                                fe898eced21b")]
public class TaskPlugin :
GoogleDesktopDisplayPluginHelperClass,
IGoogleDesktopDisplayPluginHandler,
IObjectWithSite,
IPersistStreamInit
{
static String controlGuid =
    "{766bcd9b-0b64-469c-b396-fe898eced21b}";
static String pluginName = "ioProgrammo - Task
                                          Plugin";
static String pluginDesc = "<Description>";

      ...
}
```

Dichiariamo anche a livello di classe delle variabili private che utilizzeremo poi nella registrazione e nella deregistrazione del plugin nella sidebar. Nel codice riportato occorre evidenziare l'uso l'attributo Guid che consente di registrare il plugin come oggetto COM in maniera univoca. Il primo passo da fare è la creazione dei metodi per caricare, salvare e inizializzare le impostazioni del plugin. Per questo scopo ci viene incontro l'interfaccia *IPersistStreamInit* in particolare con i metodi *Load*, *Save* e *InitNew*. Il metodo *InitNew* richiama il metodo *Initialize* che esegue le impostazioni di base del plugin:

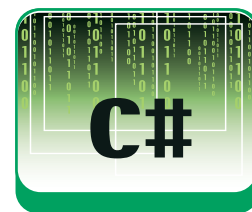
```
private void Initialize()
{
IGoogleDesktopDisplayPluginHelper helper =
(IGoogleDesktopDisplayPluginHelper)this;
GoogleDesktopDisplayContentFlags contentFlags =
GoogleDesktopDisplayContentFlags.GDD_CONTENT_
                               LAG_HAVE_DETAILS;
helper.SetFlags(GoogleDesktopDisplayPluginFlags.G
                             D_PLUGIN_FLAG_NONE,
contentFlags);
   UpdateContent();
}
```

il metodo Initialize esegue l'impostazione della modalità di visualizzazione del plugin come l'indicazione che deve poter mostrare i dettagli di ogni item. Inoltre esegue il metodo UpdateContent che si occupa di caricare gli elementi all'interno del plugin. Prima di eseguire l'implementazione del metodo, impostiamo i metodi richiesti dall'interfaccia *IObjectWithSite* dove viene passato un riferimento al cosiddetto sito del contenitore del plugin:

```
IGoogleDesktopDisplaySite displaySite;
...
public void SetSite(object site)
{
displaySite = (IGoogleDesktopDisplaySite)site;
if (displaySite != null)
Start();
else
Stop();
}
```

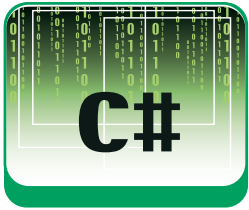
in questo modo, dopo aver dichiarato a livello di classe la variabile *displaySite* di tipo *IGoogleDesktopDisplaySite*, utilizziamo il metodo *SetSite* anche per avviare il polling del plugin che si occuperà di verificare ad intervalli prestabiliti di 60 secondi, utilizzando un oggetto timer, la presenza di nuovi task nel database:

```
private void Start()
{
timer = new Timer();
timer.Tick += new EventHandler(OnTick);
timer.Interval = 60000;
timer.Start();
}
```

il metodo *OnTick*, richiamato dall'evento Tick dell'oggetto timer, esegue una chiamata al metodo *UpdateContent*, il vero fulcro del plugin. In esso, infatti, viene effettuata l'interrogazione del database ed estratti i task da visualizzare. Il metodo utilizza l'oggetto *Task*

PERSISTENZA DELLO STATO DEL PLUGIN

IPersistStream permette ad un client di chiedere ad un oggetto di caricare i suoi dati persistenti e salvarli in uno stream. In genere quest'interfaccia è supportata da oggetti che hanno dati piuttosto semplici e che possono essere quindi memorizzati in un singolo stream. Quest'interfaccia ha solo cinque metodi:

- **Load chiede all'oggetto di caricare i suoi dati persistenti dallo stream di cui il client fornisce la posizione.**
- **Save chiede all'oggetto di salvare i suoi dati persistenti nello stream indicato dall'oggetto.**
- **IsDirty permette al client di sapere se i dati dell'oggetto sono stati modificati e se pertanto devono essere salvati prima di rilasciare l'oggetto.**
- **GetSizeMax restituisce la massima dimensione dello stream nel caso in cui il client volesse chiamare subito il metodo Save.**

IPersistStreamInit eredita tutti i metodi da IPersistStream, ma ha in aggiunta un ulteriore metodo, chiamato InitNew, con il quale il client fa sapere all'oggetto che sta per essere inizializzato per la prima volta.

ContentItem come contenitore dei vari task recuperati:

```
void OnTick(object sender, EventArgs e)
{
        UpdateContent();
}

void UpdateContent()
{

// qui utilizzo un dataset tipizzato
// ed un TableAdapter tipizzato
// per recuperare le informazioni dal database

Archivio.MyData myData = new Archivio.MyData();
Archivio.MyDataTableAdapters.TasksTableAdapter
adapter =
new
 TaskPlugin.Archivio.MyDataTableAdapters.TasksTableA
                                         dapter();
adapter.Fill(myData.Tasks);

// qui imposto un flag comune a tutti
// i ContentItem che consente di
// visualizzare un'icona di fianco ad ogni item

GoogleDesktopDisplayContentItemFlags itemFlags =
GoogleDesktopDisplayContentItemFlags.GDD_CONT
                      NT_ITEM_FLAG_LEFT_ICON;

// rimuovo tutti gli item visualizzati

this.RemoveAllContentItems();
foreach (Archivio.MyData.TasksRow row in
                              myData.Tasks.Rows)
{

// imposto le variabili relative ad
// ogni singolo ContentItem

String heading = row.title + " " +
          row.date.ToString("dd/MM/yyyy HH:mm");
String source = "MyTaskList";
String snippet = row.description;

// inizializzo l'oggetto corrente
TaskContentItem curItem = new TaskContentItem();

// imposto le proprietà del ContentItem corrente
curItem.heading = heading;
curItem.tooltip = heading;
curItem.source = source;
curItem.snippet = snippet;
curItem.flags = itemFlags;
curItem.image = contentIcon1;

// imposto il layout del ContentItem
curItem.layout =
 GoogleDesktopDisplayContentItemLayout.GDD_CONT
                      ENT_ITEM_LAYOUT_EMAIL;

// Visualizzo l'item nella sidebar e, opzionalmente,
// nella notifier window se la sidebar è nascosta
GoogleDesktopContentItemDisplayOptions options =
GoogleDesktopContentItemDisplayOptions.GDD_ITE
                         _DISPLAY_IN_SIDEBAR |
GoogleDesktopContentItemDisplayOptions.GDD_ITE
  _DISPLAY_AS_NOTIFICATION_IF_SIDEBAR_HIDDEN;

// aggiungo il ContentItem creato
this.AddContentItem(curItem, options);
}
}
```

il codice visualizzato recupera i dati dal database ed esegue un ciclo tra i record trovati. Per ogni record viene creata un'istanza dell'oggetto TaskContentItem che si occupa di visualizzare il contenuto impostato sulla base del layout impostato.
Vengono inoltre impostate le opzioni che con-

GoogleDesktopContentItemDisplayOptions
Analizziamo le diverse impostazioni possibili per il comportamento che ogni singolo ContentItem può avere con la sidebar di Google Desktop:

GDD_ITEM_DISPLAY_IN_SIDEBAR	**Consente di visualizzare un singolo ContentItem nella sidebar**
GDD_ITEM_DISPLAY_IN_SIDEBAR_IF_VISIBLE	**Consente di visualizzare un singolo ContentItem nella sidebar se questa è visibile**
GDD_ITEM_DISPLAY_AS_NOTIFICATION	**Consente di visualizzare una notifica nel momento in cui viene aggiunto il ContentItem**
GDD_ITEM_DISPLAY_AS_NOTIFICATION_IF_SIDEBAR_HIDDEN	**Consente di visualizzare una notifica nel momento in cui viene aggiunto il ContentItem se la sidebar è nascosta**

GoogleDesktopDisplayContentItemLayout
Google Desktop Display permette di utilizzare diverse modalità di layout per i singoli ContentItem. In questa tabella riassumiamo le varie possibilità consentite:

GDD_CONTENT_ITEM_LAYOUT_NOWRAP_ITEMS	**Visualizza una lista normale in cui ogni ContentItem occupa esclusivamente una singola riga**
GDD_CONTENT_ITEM_LAYOUT_NEWS	**Ogni singolo ContentItem occupa due righe composte da un'intestazione e dalla data di creazione dello stesso**
GDD_CONTENT_ITEM_LAYOUT_EMAIL	**Vengono visualizzate un'intestazione, il sorgente del ContentItem e due righe di abstract**

sentono all'item di essere visualizzato nella sidebar e nella notification area se la sidebar è nascosta. Il codice completo, compreso il database e l'accesso al database, è presente sul CD allegato alla rivista.

IL CONTENT ITEM

Ogni plugin è costituito da almeno un content item che si occupa di visualizzare i dati relativi ad un singolo elemento. Nel nostro caso, infatti, il content item visualizzerà ognuno dei task presenti nel database mostrando all'interno del plugin un breve elenco degli stessi. Aggiungiamo al progetto una nuova classe e la chiamiamo TaskContentItem.cs. Inseriamo il riferimento al namespace *GoogleDesktopDisplayLib*, ereditiamo dalla classe *GoogleDesktopDisplayContent ItemHelperClass* ed infine implementiamo l'interfaccia *IGoogleDesktopDisplayContentItem Handler*. Ereditando dalla classe helper possiamo usufruire di molte delle caratteristiche già implementate come, ad esempio, i layout standard. Per questo motivo implementiamo solo i metodi richiesti dall'interfaccia *IGoogleDesktop Display ContentItemHandler*:

```
public interface
        IGoogleDesktopDisplayContentItemHandler
{
void DrawItem(GoogleDesktopDisplayTarget target,
                    int dc, ref tagRECT bounds);
int GetHeight(GoogleDesktopDisplayTarget
                display_target, int dc, int width);
void
    GetIsTooltipRequired(GoogleDesktopDisplayTarget
        target, int dc, ref tagRECT bounds, out bool
                                        is_required);
bool OnDetailsView(out string title, out
    GoogleDesktopDisplayDetailsViewFlags flags, out
                            object details_control);
bool OnRemoveItem();
void OpenItem();
voidProcessDetailsViewFeedback(GoogleDesktopDisp
                    ayDetailsViewFlags flags);
void ToggleItemPinnedState();
}
```

Come detto, tutto il lavoro di visualizzazione del contenuto viene svolto dalla classe base *GoogleDesktopDisplayContentItemHelperClass* che si occupa, appunto, di elaborare le impostazioni da noi date nel metodo *UpdateContent* della classe *TaskPlugin* per costruire il layout, in riferimento all'impostazione scelta settando la proprietà layout. Se, opzionalmente, il nostro *ContentItem* deve poter visualizzare un dettaglio in riferimento ad una singola selezione, allora dobbiamo implementare il metodo *OnDetails View* che si occupa di gestire l'evento click associato alla visualizzazione del dettaglio:

```
public new bool OnDetailsView(out string title,
out GoogleDesktopDisplayDetailsViewFlags flags,
out object detailsControl)
{
TaskDetailsView details = new TaskDetailsView();
details.Description = snippet;
title = heading;
flags =
 GoogleDesktopDisplayDetailsViewFlags.GDD_DETAILS
                                _VIEW_FLAG_NONE;
detailsControl = details;
return false;
}
```

il metodo crea semplicemente un'istanza del controllo *TaskDetailsView*, uno user control che si occupa di fornire un'interfaccia personalizzata per la visualizzazione dei dettagli. Parliamo di interfaccia personalizzata perché è anche possi-

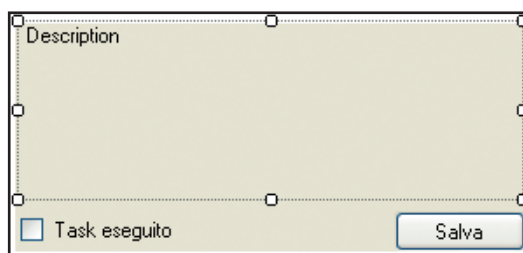
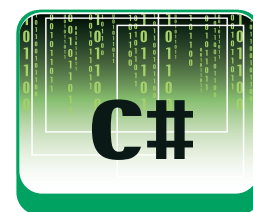
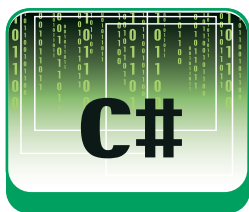



Fig. 2: ***La form per il gadget***

GoogleDesktopDisplayContentItemFlags
Flags che inidicano come un particolare ContentItem dovrebbe essere visualizzato:

Flag	Descrizione
GDD_CONTENT_ ITEM_FLAG_NONE	Nessun comportamento
GDD_CONTENT_ ITEM_FLAG_STATIC	Imposta il ContentItem come non cliccabile
GDD_CONTENT_ ITEM_FLAG_HIGHLIGHTED	Evidenzia il ContentItem
GDD_CONTENT_ ITEM_FLAG_PINNED	Blocca il ContentItem
GDD_CONTENT_ITEM_ FLAG_TIME_ABSOLUTE	Visualizza il datetime del ContentItem come datetime assoluto
GDD_CONTENT_ITEM_ FLAG_NEGATIVE_FEEDBACK	Consente agli utenti di fare commenti negative sul ContentItem
GDD_CONTENT_ITEM_ FLAG_LEFT_ICON	Permette di visualizzare una icona sul lato sinistro del ContentItem
GDD_CONTENT_ITEM_ FLAG_NO_REMOVE	Non è possibile rimuovere il ContentItem

bile utilizzare le funzionalità standard ed il layout standard di Google Sidebar semplicemente istanziando la classe *GoogleDesktop Display DetailsViewHelper*. La visualizzazione a design-time del nostro controllo sarà simile a quella in figura 2

COMPILAZIONE ED INSTALLAZIONE DEL PLUGIN

La Google Sidebar interagisce con oggetti di tipo COM, perciò dobbiamo impostare il progetto per esporre il nostro plugin come oggetto COM. Facendo click con il tasto destro sul progetto, apriamo le proprietà e ci posizioniamo nella prima scheda Application. Qui clicchiamo sul tasto Assembly Information, selezioniamo Make Assembly COM-Visible e confermiamo per rendere visibile il nostro assembly verso le applicazioni COM. Ora posizioniamoci nella scheda Build e verifichiamo che il checkbox Register for COM Interop sia selezionato. In questo modo consentiamo a Visual Studio .NET di eseguire la registrazione dell'assembly come oggetto COM, utilizzando il tool regasm.exe. Dopo aver correttamente eseguito queste impostazioni, è necessario installare il plugin per la visualizzazione nella Google Sidebar. Per far questo apriamo il file TaskPlugin.cs e nella classe *TaskPlugin* inseriamo i metodi *RegisterFunction* e *Unregister Function* che, decorati rispettivamente con gli attributi *ComRegisterFunction* Attribute e *ComUnregisterFunctionAttribute*, vengono richiamati al momento della registrazione con regasm.exe. Il codice completo dei due metodi è riportato sul CD allegato alla rivista. Provando a compilare il progetto dovrebbe comparirci un messaggio simile a quello visualizzato in figura 3 –

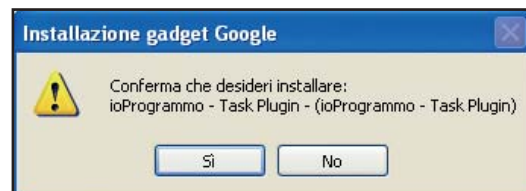



Fig. 3: *La finestra che chiede conferma per l'indicazione*

Ora il controllo è stato installato nella google sidebar e possiamo visualizzare il risultato così come riportato in figura 4

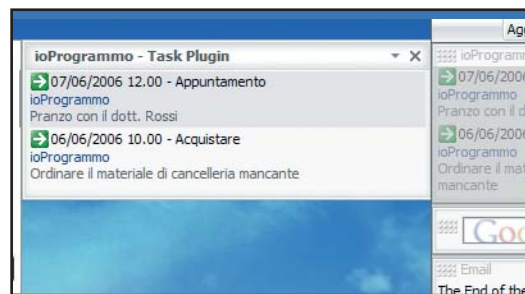



Fig. 4: *Il nostro plugin in azione*

L'AUTORE

Fabio Cozzolino sviluppa software in da circa 10 anni. Si è appassionato al .NET Framework fin dalla prima versione sviluppando principalmente applicazioni web, raggiungendo la certificazione MCAD. Negli ultimi due anni si è dedicato alla produzione di software in architetture distribuite e service-oriented. E' cofondatore dello user group DotNetSide con il quale organizza eventi periodici dedicati al mondo .NET.

UNA PROPERTY WINDOW PER IL PLUGIN

Google Desktop Display fornisce anche le API per la realizzazione di una pagina di proprietà della nostra sidebar. Per farlo seguiamo pochi semplici passi:

1. **Aggiungiamo al nostro progetto un nuovo form e lo chiamiamo PropertyForm**
2. **Oltre al codice per ereditare dalla classe base Form, aggiungiamo anche l'implementazione dell'interfaccia COM IPropertyPage presente nel file imports.cs**
3. **Tra i metodi dell'interfaccia IPropertyPage utilizziamo l'implementazione del metodo Apply() per salvare le impostazioni del nostro plugin. L'implementazione dei metodi è presente nel codice sul CD allegato alla rivista**
4. **Progettiamo il form utilizzando il designer di Visual Studio .NET, come in figura – PropertyPage**
5. **Apriamo il sorgente del nostro plugin TaskPlugin.cs**
6. **Aggiungiamo alla classe l'interfaccia**

ISpecifyPropertyPages costituita da un solo metodo GetPages:

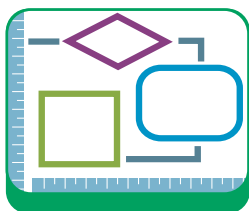
```
public void GetPages(ref CAUUID pages)
{
    Guid[] g = new Guid[1];
    g[0] = typeof(PropertyPage).GUID;
    pages.SetPages(g);
}
```

CONCLUSIONI

Nell'articolo abbiamo visto come è possibile creare un plugin per Google Desktop, utilizzando l'SDK e gli oggetti COM esposti, anche in C# con Visual Studio .NET. Seguendo alcuni passaggi abbiamo dapprima creato il plugin, poi il suo *ContentItem* ed infine il *DisplayDetails* che visualizza il singolo dettaglio all'interno di un comune *UserControl*. Anche se non è possibile sfruttare tutte le caratteristiche dell'ambiente managed, possiamo utilizzare il designer di Visual Studio .NET per progettare il nostro controllo in maniera visuale. la progettazione non è certamente delle più semplici, tuttavia una volta assimilati i concetti di base è possibile ottenere risultati sorprendenti. D'altra parte veicolare una propria applicazione attraverso un container come Google Desktop Search, è indice di sicuro successo.

Fabio Cozzolino

ALLA SCOPERTA DEL PATTERN OBSERVER

I PATTERN SONO "SOLUZIONI" UNIVERSALI A PROBLEMI COMUNI. IN QUESTO NUMERO AFFRONTIAMO IL CASO IN CUI UN OGGETTO DEVE COMUNICARE IL SUO STATO AD ALTRI. COME FARLO IN MODO DA GARANTIRE UN CODICE FACILMENTE ESPANDIBILE?

Nei vari settori della tecnologia si affermano, col passare del tempo e con l'accumularsi dell'esperienza, "tecniche" di lavoro. Queste tecniche sono un sunto delle "esperienze sul campo" e si affiancano alla conoscenza teorica che ogni persona può avere nel proprio campo di competenza. Ad esempio è molto probabile che un neolaureato in medicina abbia concetti teorici più "freschi" rispetto a un medico con 10 anni di esperienza, tuttavia credo chiunque preferirebbe farsi curare da quest'ultimo. Il settore della programmazione non sfugge a questo principio e, anzi, ne riesce a formalizzare i metodi. Questo consente ai volenterosi, di apprendere rapidamente concetti che altrimenti richiederebbero anni di esperienza per essere assimilati. Quante volte ci è capitato di potere venire a capo di problemi apparentemente molto diversi tra loro, attraverso la medesima soluzione? Per chi scrive codice tutti i giorni questa è una evenienza abbastanza frequente. Consci di questo fatto un gruppo di programmatori professionisti e studiosi del settore ha deciso di pubblicare, nel 1995, un libro che è presto diventato un riferimento per molti. Si tratta di "Design Patterns: Elements of Reusable Object-Oriented Software" e gli autori sono la cosiddetta "Gang of Four", i "quattro amici" che sono diventati per molti una sorta di "guida spirituale" in ambito di programmazione (si tratta di Gamma, Helm, Johnson e Vlissides).

REQUISITI

Conoscenze richieste
Basi di C++

Software
Un qualunque compilatore C++

PROGRAMMARE CON I PATTERN

Che cosa sono i pattern? E in che modo possono aiutarci a scrivere codice migliore? I pattern sono sostanzialmente delle soluzioni standard a delle problematiche frequenti in campo software. Queste soluzioni sono inserite all'interno di un contesto nel quale viene spiegata la problematica, viene spiegata la soluzione al problema e come ci si è arrivati, e viene fornito un esempio pratico di applicazione della soluzione. L'insieme di questi elementi costituisce quindi un "percorso" che il programmatore può seguire per giungere a una soluzione che sia garanzia di affidabilità, in quanto applicata (e testata) più volte in altri ambiti.
I pattern della Gang of Four si basano pesantemente sulla programmazione orientata agli oggetti (OOP) e mirano a migliorare la qualità del codice agendo su caratteristiche legate ad essa. Ad esempio mirano ad aumentare la riusabilità del codice, ad aumentare il livello di incapsulamento delle informazioni di un oggetto e a diminuire il livello di coesione tra oggetti di classi differenti.
Questo modo di operare porta alla costituzione di una serie di "mattoncini" di codice riutilizzabili nelle situazioni più diverse, operando solamente un minimo (o anche nullo) riadattamento.
È possibile leggere un libro sui pattern (ne esistono diversi) utilizzandolo come una sorta di "catalogo delle soluzioni". Ogni pattern indirizza una problematica ben distinta e ciascun pattern è più o meno marcatamente utilizzabile in modo separato da qualunque altro. Descriveremo di seguito nell'articolo il pattern chiamato "Observer" che è di sicuro uno dei più utilizzati e più utili quando si ha a che fare con la sincronizzazione tra due o più oggetti in una applicazione.

IL PATTERN OBSERVER

Abbiamo detto che ciascun pattern indirizza una problematica ben precisa e la soluzione a questa problematica è ottenuta per gradi. Analizziamo ad esempio la seguente situazione:

- abbiamo una applicazione dove diversi oggetti devono mantenere una informazione aggiornata sullo stato di altri oggetti;
- alcuni oggetti producono dei dati, altri devono essere informati sui dati prodotti in maniera tale da elaborarli.

Un caso concreto potrebbe essere ad esempio quel-

lo di un software che equipaggia una bici da camera elettronica. Gli oggetti *Cardio e Bike* producono dati sul battito cardiaco di chi pedala e dati sull'utilizzo della bicicletta, come ad esempio il numero di pedalate al minuto. L'oggetto *CalcolaCalorie* ha bisogno di essere informato sui dati prodotti per elaborare il consumo medio di calorie da parte dell'atleta.
Dal punto di vista della programmazione il problema consiste nel fare sì che un oggetto (*ContaCalorie*) possa essere informato sul cambiamento di stato di un altro oggetto (*Cardio* o *Bike*). *Cardio* e *Bike* sono i soggetti (*Subject*) dell'osservazione, che viene compiuta da *ContaCalorie*, che è l'osservatore (*Observer*).
Come implementare la notifica del cambiamento di stato da parte dei due produttori? Una possibile soluzione potrebbe essere la seguente:

```
class ContaCalorie
{
public:
            void ImpostaCardioFrequenza( /* ... */ );
            void ImpostaPedalate( /* ... */ );

            /* Altre funzioni qui ... /*

};

class Cardio
{
public:
            void ImpostaRiferimento(const
                                    ContaCalorie*);
            /* Altri metodi pubblici */

private:
            ContaCalorie* m_riferimento_contacalorie;
};
class Bike
{
public:
            void ImpostaRiferimento(const
                                   ContaCalorie*);
            /* Altri metodi pubblici */
private:
            ContaCalorie* m_riferimento_contacalorie;
};
```

In questo codice il meccanismo di funzionamento prevede che ogni subject mantenga un riferimento al suo observer. Il riferimento all'observer deve essere impostato nei subject dall'esterno tramite la funzione *Imposta Riferimento()*. Questa funzione valorizza il campo privato *m_riferimento_contacalorie*. Ogni qual volta lo stato del subject cambia, questi si prende carico di informare il suo observer (o i suoi observer), chiamando le relative funzioni. Ad esempio:

```
m_riferimento_contacalorie
                    >ImpostaCardioFrequenza( /* ... */);
...
m_riferimento_contacalorie->ImpostaPedalate( /* ...
                                                 */ );
```

Questa è una possibile soluzione del problema. Nel prossimo paragrafo spieghiamo perché sia una delle peggiori soluzioni adottabili.

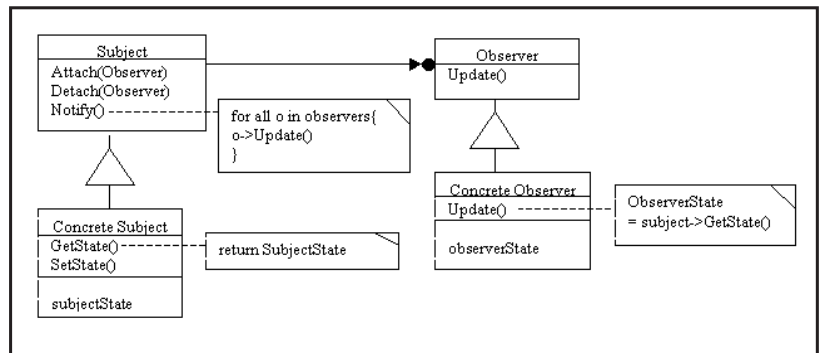
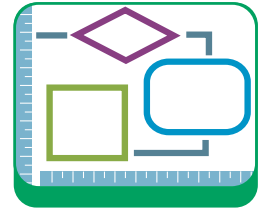
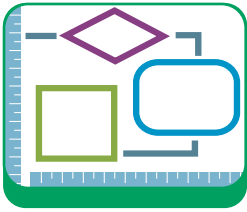



Fig. 1: **GUI dell'applicazione di esempio vista dal Flow Designer**

COSA C'È CHE NON VA

Il codice appena visto indubbiamente ha il pregio di essere funzionante. Tuttavia per chi si prefigge di scrivere codice di qualità, il funzionamento di un software è il punto di partenza, non il traguardo (in altre parole: "almeno deve funzionare!").
Perché questa soluzione ci fa tanto, tanto storcere il naso? La risposta è: perché è monolitica. Dovessimo cambiare idea su una sola parte del funzionamento globale del programma, dovremmo riscrivere da capo praticamente tutto.
Le classi *Cardio* e *Bike* presentano un'alta coesione con l'oggetto *ContaCalorie*. Probabilmente *ContaCalorie* non avrebbe nemmeno senso di esistere se non in accoppiata con i suoi *subject*. L'alta coesione fra classi è sempre una cosa che va evitata poiché per piccole modifiche costringe a riscrivere (e quindi buttare via) porzioni di codice molto grandi. Un altro problema abbastanza grave che risiede in questo codice è di ordine strettamente concettuale. Perché devono essere i soggetti dell'osservazione a dovere inviare informazioni agli oggetti che desiderano osservarli?
Come si fa a sapere se in futuro ci saranno o meno altri tipi di osservatori? In tal caso è evidente che si dovrà riscrivere molto codice, perdendo tempo e introducendo possibili bug.
Supponiamo ad esempio di volere aggiungere una nuova classe *DistanzaPercorsa* come elaboratore dei

dati forniti da *Bike* e *Cardio*. Dovremmo innanzitutto scrivere una classe con una struttura pressoché identica a *ContaCalorie*:

```
class DistanzaPercorsa
{
public:
        void ImpostaCardioFrequenza( /* ... */ );
        void ImpostaPedalate( /* ... */ );
        /* Altre funzioni qui ... /*
};
```

questo viene meno al principio di "incapsulare le informazioni che cambiano" e si traduce nella proliferazione di tanto codice che si ripete identico in diversi punti.
Successivamente dovremmo per forza modificare le classi subject in modo che esse possano contenere il riferimento al nuovo observer, e in modo tale che, ogni qual volta il loro stato cambia, anche questo nuovo observer ne sia informato:

```
m_riferimento_distanza->ImpostaCardioFrequenza(
                                        /* ... */ );
m_riferimento_distanza->ImpostaPedalate();
```

Se pensiamo quindi al fatto che per aggiungere un semplice observer dobbiamo obbligatoriamente modificare il codice di tutti i subject cui esso è interessato, capiamo bene come il numero delle modifiche, col crescere degli observer, diventerebbe presto ingestibile.
Il tutto senza contare la presenza di altre grosse limitazioni. Se ad esempio volessimo inserire più di un oggetto per ogni classe observer (ad esempio mettere due oggetti *Conta Calorie*), semplicemente non potremmo farlo, in quanto ogni subject contiene un solo riferimento come attributo privato. Inserire riferimenti in un array contribuirebbe ulteriormente ad aumentare il numero di modifiche al codice in caso di cambiamenti anche piccolissimi.
Insomma: soluzione scartata!

PICCOLA BIBLIOGRAFIA SUI PATTERN.

Il libro che ha contribuito alla formalizzazione del concetto di "pattern" e che rappresenta il capostipite di questa letteratura è sicuramente quello citato nel corpo dell'articolo e cioè:

- **"Design Patterns: Elements of Reusable Object-Oriented Software" (Gamma, Helm, Johnson, Vlissides).**

Questo libro tuttavia, oltre a definire dei pattern, definisce anche una maniera standard per formalizzare un pattern. Anche grazie a questo fatto sono stati scritti diversi altri libri sui pattern, applicati in ambiti differenti. Ad esempio segnaliamo:

- **"Design Patterns Java Workbook" (Steven John Metsker)**
- **"Pattern Hatching: Design Patterns Applied" (John Vlissides)**
- **"Framework-based Software Development in C++" (Gregory F. Rogers)**
- **"Patterns and Skeletons for Parallel and Distributed Computing" (F.A. Rabhi and S. Gorlatch)**
- **"Small Memory Software: Patterns for Systems with Limited Memory" (James Noble & Charles Weir)**

UNA SOLUZIONE MIGLIORE

Per ottenere qualcosa di qualitativamente migliore, che sia riutilizzabile senza troppi sforzi, dobbiamo necessariamente ripensare l'approccio intrapreso in precedenza. Innanzitutto va eliminata la dipendenza del subject dal suo observer. L'oggetto osservato deve avere come unico compito quello di segnalare che il suo stato è cambiato. Non deve essere tuttavia obbligato a conoscere quale sia il suo observer (o quanti siano), né di quali dati l'observer abbia bisogno. Sarà l'observer stesso, una volta ricevuta la notifica del fatto che "qualcosa è cambiato", a richiedere le informazioni di cui ha bisogno, invocando i metodi pubblici del subject.
Questo meccanismo è mirato a realizzare il disaccoppiamento tra subject e observer. Per slegare inoltre il concetto astratto di subject e observer, dalle loro varie realizzazioni concrete (ad es. Cardio-subject, Contacalorie-observer) è opportuno utilizzare uno dei meccanismi proposti dai linguaggi basati sugli oggetti, e cioè l'ereditarietà. Quello che si fa è incapsulare il codice generico di un subject o di un observer in una classe base.
Da questa classe deriveremo le nostre classi specializzate, che forniranno le funzionalità di cui abbiamo bisogno, ma utilizzeranno il codice delle loro classi base per implementare il meccanismo di notifica e richiesta dello stato.
Chiamando Subject e Observer le classi base, indichiamo genericamente con *ConcreteSubject* e *ConcreteObserver* le loro derivate.
Sostituiremo in seguito queste realizzazioni con le classi previste nel nostro esempio.

```
class Subject
{
public:
        virtual void Attach(Observer*);
        virtual void Detach(Observer*);
        virtual void Notify();
        /* Altri eventuali metodi qui ... */
};
class Observer
{
public:
        virtual void Update();

        /* Altri eventuali metodi qui ... */
};
```

Il diagramma UML completo per il pattern Observer è mostrato in FIGURA. Il funzionamento è molto semplice:

- l'Observer, per notificare al Subject di essere interessato al suo stato, si connette ad esso tramite la funzione *Attach*() (la funzione *Detach*() esegue l'operazione inversa);
- ogniqualvolta un Subject vuole notificare ai suoi Observer un suo cambiamento di stato chiama la funzione *Notify*(). Questa funzione richiama a sua volta la Update() per tutti gli Observer connessi;
- il codice della funzione *Update*() deve essere specializzato nella classe che deriveremo da Observer. Per questo motivo dichiariamo tale funzione con la parola chiave "virtual".

Nel corpo della *Update*() tipicamente si inserisce il codice per richiedere al Subject le informazioni sul suo stato. Anche questo tipo di richiesta varia a seconda di come specializziamo le derivate di Subject. Vediamo come effettuare queste specializzazioni.

LA SPECIALIZZAZIONE

La struttura di classi appena presentata deve essere utilizzata come base per derivare classi che modelleranno il comportamento da noi voluto. Questo è un grande pregio poiché il codice scritto per Observer e Subject resterà lo stesso anche quando in futuro vorremo utilizzare il pattern Observer per una applicazione completamente differente. È questo che si intende per riusabilità del codice.

Le nuove classi *Cardio*, *Bike* e *ContaCalorie* sono dichiarate come segue:

```
class Cardio : public Subject
{
public:
        int GetCardioFrequenza();

        /* Altri eventuali metodi qui ... */
};

class Bike : public Subject
{
public:
        int GetPedalate();

        /* Altri eventuali metodi qui ... */
};

class ContaCalorie : public Observer
{
public:
        virtual void Update();

        /* Altri eventuali metodi qui ... */
};
```

Come si vede la specializzazione consiste nell'inserire i metodi specifici per gli oggetti che ci interessano. In particolare abbiamo fornito l'interfaccia di Cardio e Bike delle relative funzioni GetX(), funzioni che nella nostra prima implementazione-di-scarsa-qualità erano poste, in maniera innaturale, all'interno dell'oggetto che osservava, anziché di quello che era osservato.

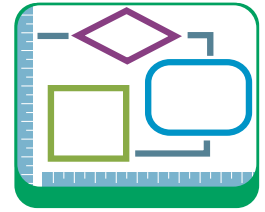
Per quanto riguarda la funzione Update() una possibile struttura di codice potrebbe essere la seguente:

```
// supponiamo di avere memorizzato i riferimenti ai
                                              subject
// in apposite variabili membro di ContaCalorie

void ContaCalorie::Update()
{
int pedalate = m_riferimento_bike->GetPedalate();
this->ElaboraPedalate(pedalate);

int frequenza = m_riferimento_cardio
                              >GetCardioFrequenza();
this->ElaboraCardioFrequenza(frequenza);
}
```

CONCLUSIONI

In una complessa applicazione orientata agli oggetti, molti di questi lavorano all'unisono per ottenere il comportamento desiderato. In questo contesto la gestione delle dipendenze tra gli stati di questi oggetti è un problema fondamentale. In questo articolo abbiamo mostrato come il pattern Observer possa essere utilizzato per mantenere la consistenza tra gli stati. Lo abbiamo fatto partendo da un caso concreto e mostrandole differenze tra una soluzione rabberciata e una di qualità.

L'utilizzo dei pattern può rappresentare sicuramente un miglioramento della qualità del codice che scriviamo. Tuttavia non dobbiamo considerarlo come la "manna dal cielo" che risolve tutti i problemi. Inoltre lo studio dei pattern, per quanto serva a dare un notevole balzo in avanti nell'esperienza del design delle applicazioni, non si sostituisce (né mai alcuna cosa potrà farlo) alla reale esperienza sul campo, per cui a volte è bene ascoltare il parere di qualche collega con magari qualche anno di esperienza in più e qualche libro sui pattern in meno sugli scaffali di casa. Insomma anche per usare correttamente i pattern c'è bisogno di esperienza!

Alfredo Marroccelli

CREARE APPLICAZIONI J2ME CON NETBEANS

LE APPLICAZIONI PER DISPOSITIVI MOBILI SU PIATTAFORMA J2ME RAPPRESENTANO UNA PERCENTUALE ELEVATA FRA QUELLE SUL MERCATO. SCOPRIAMO COME CREARLE IN MODO FACILE E VELOCE ATTRAVERSO L'USO DEL MOBILITY PACK DI NETBEANS

Il mercato degli ambienti di sviluppo di supporto alla programmazione è in continua evoluzione. Sicuramente la tecnologia Java vanta il maggior numero di tool facenti parte di questa categoria. Tra gli strumenti maggiormente diffusi si possono citare Eclipse, Netbeans, JBuilder, JDeveloper e IntelliJ IDEA. Ciascun tool ha i propri punti di forza che lo caratterizzano e fanno sì che venga scelto come IDE di riferimento. Dopo l'introduzione della piattaforma J2ME per dispositivi mobili, tutti i gruppi di sviluppo IDE si sono messi a lavoro per supportare ed agevolare i programmatori nell'implementazione delle nuove applicazioni per questo tipo di tecnologia.
Alcuni produttori hanno scelto di integrare in modo inscindibile il supporto allo sviluppo di applicazioni mobili all'IDE stesso; altri, come Netbeans o Eclipse, hanno preferito creare moduli separati (add-on o plugin) per sopperire a tale mancanza. Nel panorama J2ME il Mobility Pack, un add-on del Netbeans IDE distribuito gratuitamente, risulta essere uno degli strumenti maggiormente utilizzati in quanto presenta delle caratteristiche innovative che permettono di sviluppare applicazioni in modo fluido e veloce. Il componente più interessante, su cui ci soffermeremo nel corso dell'articolo, è il Visual Mobile Designer che, così come il Matisse GUI Builder per la creazione di interfacce Swing, permette di creare e modificare in maniera visuale (via drag and drop) le componenti grafiche per applicazioni costruite sullo strato MIDP 2.0.

UN'APPLICAZIONE DI ESEMPIO

In questo articolo, per meglio capire il funzionamento e le potenzialità del Mobility Pack, implementeremo una semplice applicazione di conversione valuta, che chiameremo MobileExchange, che dovrà presentare i seguenti requisiti funzionali:

1. screenshot di avvio con visualizzazione di un'immagine
2. caricamento da file della lista valute con i rispettivi rapporti di cambio
3. calcolo, da un importo in valuta x, del corrispettivo in valuta y

Il file contenente i rapporti di cambio non è altro che un semplice documento di testo contenente una lista di nomi di valute ed il rispettivo valore di cambio separati dal simbolo "=":

```
# rapporti cambio aggiornati al 30/05/06
Euro=1
Dollaro USA=0.7772
Sterlina=1.4629
Yen=0.6927
Franco Svizzero=0.6417
Dollaro Canadese=0.7068
Dollaro Australiano=0.5921
Pesos=0.0690
```

Per separare la parte grafica da quella dati e rendere il codice più elegante concentreremo le parti di caricamento e calcolo dei valori di cambio nella classe it.ioprogrammo.Exchange RateManager. Per garantire la presenza di una sola istanza di questo tipo all'interno della KVM (Kilo Virtual Machine) implementeremo il pattern singleton, pertanto alla prima invocazione del metodo getInstance() verrà eseguito il seguente costruttore:

```
/* Costruttore interno per l'implementazione
                              del pattern singleton */
private ExchangeRateManager() {
    // carica tutte le linee del file di valute ...
    Vector values =
        this.loadElements("/res/rates.txt", "\n");
    int size = values.size();
```

```
        this.currenciesTable = new Hashtable(size);
        String line;
        // ... popola la tabella di rapporti cambio
        for (int i = 0; i < size; i++) {
            line = (String)values.elementAt(i);
            int equalIndex = line.indexOf("=");
            // verifica il corretto inserimento della
                                 coppia <nome>=<valore>
            if (equalIndex > 0) {
            this.currenciesTable.put(line.substring(0,
                                         equalIndex),
                                new
            Double(Double.parseDouble(line.substring
                                    (equalIndex+1))));
            }
        }
    }
```

Per il calcolo dei valori di rapporto cambio esporremo il metodo getExchangeRateBetween() che necessita degli identificativi della valuta di origine e di quella finale.

```
public double getExchangeRateBetween (String
                                  cur1,String cur2) {
        if (!currenciesTable.containsKey( cur1) ||
             !currenciesTable.containsKey( cur2) ) {
             return 0.0D;
        }
        double ammount1 = ((Double
          )currenciesTable.get( cur1)).doubleValue();
        double ammount2 = ((Double
          )currenciesTable.get( cur2)).doubleValue();
        return ammount1 / ammount2;
}
```

PRIMI PASSI CON IL VISUAL DESIGNER

Adesso ci addentreremo nell'implementazione della GUI dell'applicazione attraverso il componente chiave del Mobility Pack: il Visual Mobile Designer. Questo tool, che consente il disegno e lo sviluppo di MIDlet, si divide in tre macro aree: Flow Design, Screen Design e Source. Per la creazione di una nuova Visual MIDlet cliccare su File -> New File e selezionare il tipo Visual MIDlet nella categoria MIDP. La vista ottenuta selezionando la prospettiva Flow Design consente allo sviluppatore di aggiungere degli oggetti di tipo javax.microedition. lcdui.Screen e di specificarne le connessioni tra gli stessi definendo il flusso dell'applicazione. In Figura 1 è illustrato il flusso logico che descrive la GUI dell'applicazione di esempio: la griglia centrale delinea l'area di disegno mentre nella

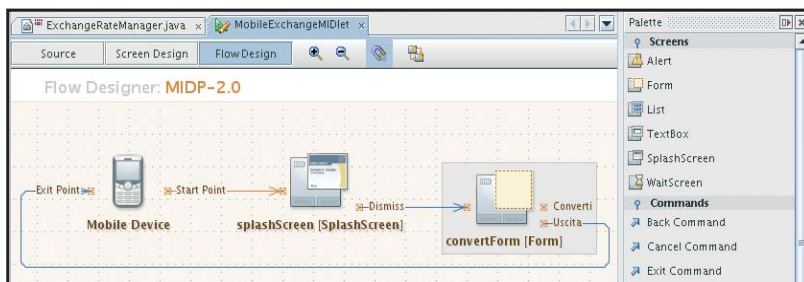



Fig. 1: *GUI dell'applicazione di esempio vista dal Flow Designer*

parte destra troviamo la palette contenente i componenti inseribili via drag and drop. L'azione successiva all'avvio dell'applicazione, rappresentata dall'etichetta "Start Point", visualizza un oggetto SplashScreen che termina con la visualizzazione della Form di conversione. Per chi ha già avuto a che fare con

COME INIZIARE

1. É indispensabile avere installato sul computer Java 2 Standard Edition 1.4 o superiore. É preferibile utilizzare l'ultima versione disponibile dal sito http://java.sun.com.

2. Scaricare Netbeans IDE ed il Mobility Pack dalla sezione Downloads del sito http://www.netbeans.org. Installare prima l'ambiente di sviluppo e successivamente il Mobility Pack. Quest'ultimo integra al suo interno la versione 2.2 del J2ME Wireless Toolkit.

3. Decomprimere su filesystem il contenuto del file netbeans.zip, contenuto nel cd-rom allegato, in una directory a piacere .

4. Una volta avviato l'IDE, cliccare su File -> Open Project e selezionare la cartella MobileExchange contenuta all'interno del file scompattato precedentemente.

5. La Visual Midlet di esempio, preparata per questo articolo, è contenuta nella classe it.ioprogrammo.MobileExchangeMIDlet. Una volta aperta sarà possibile navigare tra le modalità Source, Screen Design e Flow Design che danno la possibilità di editare il progetto sotto diversi punti di vista.

applicazioni J2ME il nome SplashScreen potrebbe suonare nuovo. Quest'ultimo, infatti, non è tra gli Screen standard messi a disposizione dallo strato MIDP ma è un elemento aggiuntivo facente parte di un set di compo-

Fig. 2: *Definizione dello Screen per la schermata di conversione valuta*

nenti grafici custom esposti dal Mobility Pack. Oltre a quello appena citato ne troviamo anche altri di grande utilità come tabelle o wait screen.
Attraverso la vista dello Screen Designer è possibile definire in maniera più dettagliata i singoli elementi Screen realizzati nello step precedente.
In Figura 2 è visualizzata la definizione del convertForm: nella parte sinistra troviamo una preview dello Screen in questione, al centro l'elenco dei comandi associati ed a destra la palette contenente oggetti Command e Item.
Come si può notare la struttura della form è abbastanza semplice ed è costruita utilizzando un TextField per l'inserimento del valore originale, due ChoiceGroup per la scelta delle valute iniziale e finale, uno StringItem per la visualizzazione del risultato e due comandi.
Per editare le caratteristiche (titolo, layout, ecc.) di ciascun elemento grafico aggiunto è necessario prima selezionarlo e poi aprire il dialog di Properties dal popup menu ottenuto cliccando sul tasto destro del mouse.

INTEGRAZIONE DEL CODICE GENERATO

Una volta completato il disegno dell'interfaccia grafica dell'applicazione non rimane che integrarla con la parte logica. Per editare il codice generato dal Designer utilizzeremo la vista Source.
Ovviamente, per non disallineare il codice generato dalle rappresentazioni di Screen e Flow Designer, non sarà possibile editare il codice integralmente. Di seguito vediamo come è stato introdotto il calcolo dell'operazione di conversione scatenata dall'azione legata al comando convert Command:

```
public void commandAction(Command
            command,Displayable displayable) {
  // Insert global pre-action code here
     if (displayable == convertForm) {
        if (command == exitCommand1) {
           // Insert pre-action code here
           exitMIDlet();
           // Insert post-action code here
        } else if (command ==
                          convertCommand) {
           // Insert pre-action code here
           // Do nothing
           // Insert post-action code here
           String value =
                       this.valueField.getString();
           // controllo la presenza del valore
           double result = 0;
           if (value != null && value.length() >
                                            0) {
              double rate =
            ExchangeRateManager.getInstance().
                         getExchangeRateBetween(
this.fromCurrencyChoice.getString
      (this.fromCurrencChoice.getSelectedIndex()),
this.toCurrencyChoice.getString
        (this.toCurrencyChoie.getSelectedIndex()));
              result =Double.parseDouble(value)
/
                                            rate;
           }
           this.stringItem1.setText(""+result);
        }
     }
  // Insert global post-action code here
}
```

Il codice in rosso evidenzia le parti che Netbeans ci vieta di modificare mentre le zone “libere” sono precedute da specifici commenti (es: // Insert post-action code here).

PRINCIPALI CARATTERISTICHE DI NETBEANS MOBILITY PACK 5.0

- **Basato su Netbeans IDE**
- **Visual Mobile Designer con editing drag and drop**
- **Possibilità di editare il codice generato dal Visual Designer**
- **Aggiunta di nuovi componenti grafici custom**
- **Offuscamento integrato, diviso in livelli, per proteggere la proprietà intellettuale ed ottimizzare la dimensione dei jar generati**
- **Preprocessor Directives per ridurre il problema di Device Fragmentation (compatibili con Antenna e J2ME Polish)**
- **J2ME Wireless Toolkit 2.2 integrato**
- **Build system Ant based**
- **Possibilità di deploy via FTP, SCP e WebDAV**
- **Supporto per la localizzazione**
- **Possibilità di aggiungere emulatori di terze parti**
- **Supporto per Java ME Web Services (JSR 172)**

PREPROCESSOR DIRECTIVES & OBFUSCATION

Tra i problemi che spesso gli sviluppatori J2ME si trovano ad affrontare ci sono quelli riguardanti il fenomeno di “device fragmentation” e le eccessive dimensioni dell'applicazione. Il primo aspetto è il risultato delle differenze implementative e della varietà di attributi che ciascun produttore apporta ai vari dispositivi provvisti di una KVM. Per minimizzare eventuali problemi di compatibilità tra i vari dispositivi è spesso necessario creare specializzazioni “ad hoc” per una serie

di vendor o addirittura famiglie di dispositivi. Questo espediente, che si traduce nell'effort di mantenere più versioni dello stesso software, è facilitato dall'uso delle Project

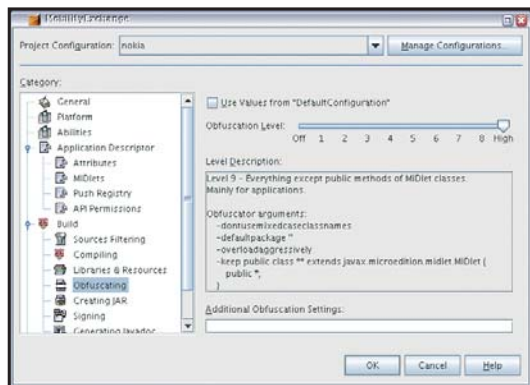



Fig. 3: ***Configurazione del livello di offuscamento***

Configurations e Preprocessor Directives messe a disposizione dal Mobility Pack. Il seguente metodo di esempio

```
1  public String greetDevice () {
2  //#if Nokia_40
3  return "Welcome Nokia 40 Series";
4  //#else
5  return "Welcome. What's your name?";
6  //#endif
7  }
```

contenente, sotto forma di commento in linea, le semplici direttive "if/else/endif", consentirà la creazione automatica di due distinte applicazioni: la prima, "assemblata" con l'invocazione return di riga 3, per dispositivi Nokia facenti parte della famiglia 40 e la seconda per tutti gli altri tipi di device.

Un'altra limitazione da tener presente è rappresentata dal "peso" dell'applicazione. Ogni dispositivo, ad eccezione di quelli di ultima generazione con possibilità di espansione di memoria, presentano delle restrizioni specificate dalla proprietà MaxJarSize che definisce la dimensione massima per un'applicazione J2ME. Per "snellire" il codice Netbeans ci offre la possibilità di offuscare il sorgente a più livelli, il tutto configuarabile dalle proprietà del progetto. Ovviamente questa possibilità risulta molto comoda, sia per ridurre il peso che per aumentare la sicurezza dell'applicazione. Per maggiori informazioni inerenti le caratteristiche e le limitazioni di ogni singolo dispositivo è possibile consultare il sito http://www.j2mepolish.org/devices-overview.html, all'interno del quale troverete informazioni dettagliate su ogni singolo dispositivo.

CONCLUSIONI

In un momento in cui lo sviluppo per dispositivi mobili rappresenta una percentuale molto elevata del mercato della programmazione, il Mobility Pack rappresenta una soluzione ad alto valore aggiunto, probabilmente l'unica in grado di contrastare la semplicità di sviluppo che si manifesta quando si sviluppa per il rivale "Pocket PC". Se J2ME vuole mantenere la leadership sul mercato dei device portatili deve affiancare alla già tradizionale flessibilità, anche strumenti di lavoro che consentano a noi programmatori di sviluppare applicazioni in modo rapido ed immediato.

In questo articolo abbiamo dimostrato come costruire una semplice applicazione di conversione valuta per dispositivi mobili in modo facile e veloce con Netbeans ed il Mobility Pack. Quest'ultimo rappresenta, al momento, lo strumento più avanzato per la creazione di applicazioni J2ME. Nei prossimi articoli introdurremo il processo di creazione di applicazioni CDC con componenti grafici avanzati. Lo scopo finale sarà quello, di mostrare da un lato come accelerare il processo di produzione, dall'altro sviluppare applicazioni complete dotate di una grafica accattivante e di funzionalità avanzate. In tutto questo ci faremo aiutare di volta in volta dai vari strumenti evoluti disponibili.

Fabrizio Fortino

COSA CI PROSPETTA IL FUTURO?

Secondo alcuni studi presentati durante l'ultima Java One Conference, tenutasi a San Francisco, oltre il 60% dei telefoni cellulari in circolazione sono equipaggiati di piattaforma Java. La grande maggioranza di questi dispositivi, a causa di limitazioni hardware, implementano lo strato CLDC (Connected Limited Device Configuration) che, come si intuisce dalla stessa sigla, mette a disposizione del programmatore un set di librerie abbastanza limitato. Con l'evoluzione dei dispositivi mobili (con memoria di almeno 2MB) sarà possibile implementare applicazioni basate su CDC (Connected Device Configuration), che presentano una API più ricca e permettono l'utilizzo di componenti grafici avanzati. Visto che il mercato della telefonia mobile è in continuo fermento, la comunità di Netbeans ha ritenuto saggio non farsi trovare spiazzata ed ha già "sfornato" il nuovo Mobility Pack 5 for CDC, disponibile per OS Windows al sito http://www.netbeans.info/downloads/download-cdc.html

Sempre durante un intervento alla Java Conference, James Gosling, ospite dell'appuntamento milanese, "padre" di Java e attuale Vice President di Sun Microsystems, ha affermato che i telefoni cellulari saranno i desktop del futuro. Questa decisa affermazione è stata accompagnata dall'ingresso nel mercato del primo dispositivo mobile dotato di un sistema operativo "aperto" realizzato in Java: Savaje (http://www.savaje.com).

Lo smartphone in questione è il Jasper S20, distribuito dalla casa americana GSPDA (http://www.gspda.com), con cui sarà possibile sfruttare le potenzialità dei componenti Swing e Java2D.

MENO CODICE CON IL NAMESPACE MY

LO SPAZIO DEI NOMI “MY” PERMETTE DI UTILIZZARE IN MODO SEMPLICE, MOLTE CLASSI DEL .NET FRAMEWORK CONSENTENDO UN’INTERAZIONE RAPIDA CON IL COMPUTER, L'APPLICAZIONE, LE IMPOSTAZIONI E LE RISORSE, RENDENDO PIÙ LEGGERO IL CODICE

Una delle più interessanti nuove caratteristiche dì Microsoft Visual Basic 2005 è l'introduzione del namespace My. Il namespace My contiene un insieme di oggetti che forniscono percorsi semplificati a molte aree del .NET Framework, riducendo il numero di righe di codice che è necessario scrivere in maniera efficiente ed affidabile. Il namespace My espone alcuni oggetti che vengono creati dinamicamente ogni volta che si aggiungono caratteristiche al progetto corrente. Gli oggetti del namespace My sono accessibili in modalità a tipizzazione forte, con l'aiuto di IntelliSense eliminando gli errori a runtime provocati da errori di digitazione. I tre oggetti My principali sono: *My.Application*, *My.Computer* e *My.User*.
Questi oggetti possono essere utilizzati per accedere alle informazioni rispettivamente sull'applicazione corrente, sul computer sul quale è installata l'applicazione o sull'utente corrente dell'applicazione. In questo articolo ci occuperemo in dettaglio dell’oggetto *My.Computer*

UNO SGUARDO DI INSIEME

Descriviamo brevemente gli oggetti contenuti nel namespace My

- *My.Application* mette a disposizione informazioni sull'applicazione corrente, come il percorso, la versione, la nazionalità e la modalità e autenticazione dell'utente.

- *My.Computer* mette a disposizione molti oggetti figli che consentono di raccogliere informazioni e utilizzare le caratteristiche principali del computer, tra cui filesystem, i sottosistemi audio e video, la memoria, la tastiera, il mouse, la rete. porte seriali, la stampante d altro.
- *My.Forms* è subito disponibile e definisce un metodo factory per ciascuna classe form definita nel progetto corrente. Permette di referenziare l'istanza di default di un form senza dover creare esplicitamente un oggetto della classe form.

- *My.Resources* contiene un oggetto per ciascuna risorsa definita nel progetto corrente. Se, ad esempio, aggiungiamo al progetto una bitmap denominata Master, sarà possibile accedervi mediante My.Resources. Master.

- *My.Settings* mette a disposizione una proprietà per ciascuna impostazione di configurazione determinata nelle impostazioni correnti; a livello di applicazione o per utente. Una risorsa a livello di applicazione può essere condivisa da tutti gli utenti, una risorsa a livello di utente può essere modificata dall'utente corrente.

- *My.User* permette di accedere alle informazioni sull'utente correntemente collegato e permette di implementare un meccanismo di autenticazione custom.

- *My.WebServices* mette a disposizione una proprietà per ciascun Web service a cui il progetto corrente ha un riferimento.
My.WebServices consente di individuare la classe in maniera automatica e fornisce un'i-

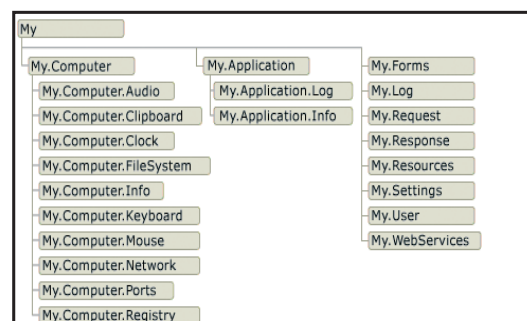



Fig. 1: La gerarchia delle classi del namespace My

stanza della classe proxy del servizio Web su richiesta.

L'OGGETTO MY.COMPUTER

L'oggetto My.Computer è un oggetto traboccante di funzionalità e fornisce le proprietà per la gestione dei componenti del computer come audio, orologio, tastiera, file system, memoria, mouse, porte seriali, la Clipboard, e il registro di sistema.
Gli oggetti messi a disposizione da My.Computer sono:
o Audio. L'oggetto My.Computer.Audio consente di accedere al sistema audio del computer e fornisce inoltre i metodi per la riproduzione dei file .wav.

- *Clipboard.* L'Oggetto My.Computer.
  Clipboard mette a disposizione i metodi per la modifica degli Appunti.
  o Clock. L'oggetto My.Computer.Clock mette a disposizione le proprietà per accedere all'ora locale corrente e al tempo universale coordinato (UTC), equivalente all'ora di Greenwich, dall'orologio di sistema.
- *FileSystem.* L'oggetto My.Computer.FileSystem mette a disposizione proprietà e metodi per l'utilizzo delle unità, dei file e delle directory.
- *Info.* L'oggetto My.Computer.Info mette a disposizione le proprietà che permettono di ottenere informazioni sulla memoria, sul nome e sul sistema operativo del computer.
- *Keyboard.* L'oggetto My.Computer.
  Keyboard mette a disposizione le proprietà per accedere allo stato corrente della tastiera, ad esempio ai tasti attualmente premuti, e un metodo per inviare le sequenze di tasti alla finestra attiva.
- *Mouse.* L'oggetto My.Computer.Mouse mette a disposizione le proprietà che consentono di ottenere informazioni sul formato e la configurazione del mouse installato sul computer locale.
- *Name.* Permette di ottenere il nome del computer.
- *Network* L'oggetto My.Computer.Network permette di accedere ai tipi e agli eventi della rete.
- *Ports* L'oggetto My.Computer.Ports mette a disposizione una proprietà e un metodo per accedere alle porte seriali del computer.
- *Registry* L'oggetto My.Computer.Registry permette di leggere e scrivere nel Registro di sistema.
- *Screen* permette di ottenere l'oggetto Screen che rappresenta lo schermo principale del computer.

L'OGGETTO MY.COMPUTER.AUDIO

L'oggetto My.Computer.Audio mette a disposizione i metodi My.Computer.Audio.Play e My.Computer.Audio.PlaySystemSound che permettono di riprodurre file audio wav e suoni del sistema.
Una caratteristica importante My.Computer. Audio è la possibilità di eseguire un file .wav, in modo sincrono oppure asincrono. Nel primo caso il metodo Play attende che venga completata l'operazione prima di passare all'istruzione successiva. Nel secondo caso l'esecuzione dell'audio avviene in background permettendo all'applicazione di eseguire altro codice mentre riproduce il suono, si può iterare l'esecuzione finché l'applicazione non esegue il metodo Stop. Utilizzando questa caratteristica si può, ad esempio, mantenere un suono in esecuzione mentre viene visualizzato un form.
Per riprodurre un suono in maniera sincrona, possiamo scrivere:

```
My.Computer.Audio.Play("C:\suono.wav",
                        AudioPlayMode.WaitToComplete)
```

Per riprodurre un suono in maniera asincrona, possiamo scrivere:

```
My.Computer.Audio.Play("C:\suono.wav",
                        AudioPlayMode.Background)
```

Il metodo Play permette la riproduzione di un solo suono in background per volta, quando l'applicazione riproduce un nuovo suono in background, interrompe il precedente.
Per riprodurre un suono ciclicamente, in background, fino a quando viene chiamato il metodo Stop, possiamo scrivere:

```
My.Computer.Audio.Play("C:\suono.wav",
                        AudioPlayMode. BackgroundLoop)
```

Con l'oggetto Audio si possono leggere informazioni audio memorizzate in uno stream o in un array di tipo Byte, però l'unico formato supportato è il formato Pulse-Code Modulation (PCM) standard; per questo motivo non è possibile riprodurre file MP3. In ogni caso questa caratteristica risulta molto comoda per eseguire piccoli suoni tesi a sottolineare un'azione o un evento dell'applicazione

MANIPOLARE LA CLIPBOARD

L'Oggetto My.Computer.Clipboard fornisce i metodi che permettono la completa gestione degli appunti. Gli appunti sono condivisi da tutti i processi attivi sul PC e per questo motivo possono essere utilizzati per memorizzare e trasferire dati, ad esempio testo, immagini e suoni tra le applicazioni. Gli elementi spostati o copiati negli appunti vengono conservati anche dopo la chiusura dell'applicazione.

I metodi che permettono di leggere i dati dagli appunti sono:

- *GetText* permette di leggere testo dagli Appunti.
- *GetImage* permette di leggere un'immagine dagli Appunti
- *GetData* permette di leggere dati dagli Appunti.
- *GetAudioStream* permette di leggere un flusso audio dagli Appunti
- *GetFileDropDownList* permette di leggere un FileDropList dagli Appunti.

Per scrivere dati negli appunti, si devono usare i metodi Set corrispondenti.
Infine per determinare quale tipo di dato si trova memorizzato negli Appunti, è possibile utilizzare i metodi: ContainsAudio, Contains FileDropList, ContainsImage e ContainsText. Per controllare un formato personalizzato, possiamo utilizzare il metodo ContainsData.

LEGGERE E SCRIVERE UN TESTO

Per scrivere dati di tipo testo negli Appunti si deve utilizzare il metodo SetText, per questo possiamo scrivere:

```
My.Computer.Clipboard.SetText("Testo da
                                memorizzare.")
```

Il metodo *SetText* ammette come parametro il tipo di enumerazione *TextDataFormat* che permette di specificare il formato del testo da scrivere. Se vogliamo, ad esempio, scrivere del testo in formato RTF:

```
My.Computer.Clipboard.SetText("Testo da
                                memorizzare.", _
System.Windows.Forms.TextDataFormat.Rtf)
```

Per leggere il testo contenuto negli appunti dobbiamo utilizzare il metodo GetText, per questo possiamo scrivere il codice seguente, in cui il testo viene letto e visualizzato in una finestra di messaggio standard

```
MessageBox.Show(My.Computer.Clipboard.GetText())
```

Per il corretto funzionamento dell'esempio, è necessario che negli Appunti sia memorizzato un testo, altrimenti viene visualizzato un messaggio vuoto.
Per cancellare il contenuto degli Appunti dobbiamo utilizzare il metodo Clear. Il metodo Clear cancella i dati di qualsiasi formato e può avere effetto sugli altri processi condivisi. Possiamo scrivere semplicemente:

```
My.Computer.Clipboard.Clear()
```

UTILIZZARE AUDIO E IMMAGINI

Per salvare i dati audio negli Appunti possiamo utilizzare il metodo:
My.Computer.Clipboard.SetAudio.
Se ad esempio vogliamo memorizzare il file audio pippo.wav dobbiamo:
creare una matrice di byte MatriceMusicale

```
Dim MatriceMusicale As Byte()
```

Assegnare a MatriceMusicale il flusso di byte contenenti l'audio, utilizzando il metodo My.Computer.FileSystem.ReadAllBytes

```
MatriceMusicale =
   My.Computer.FileSystem.ReadAllBytes("pippo.wav")
```

Memorizzare il valore della matrice negli appunti

```
My.Computer.Clipboard.SetAudio(MatriceMusicale)
```

Per leggere un'immagine dagli Appunti possiamo utilizzare il metodo:
My.Computer.Clipboard.GetImage.
Scriviamo il codice necessario a recuperare un'immagine contenuta negli appunti ed assegniamola ad una PictureBox.
Per prima cosa dobbiamo verificare che i dati contenuti negli appunti corrispondano ad una immagine, per questo motivo utilizziamo la funzione ContainsImage. Nel caso in cui gli appunti contengono un'immagine, allora l'immagine viene mostrata nella PictureBox, in caso contrario mostriamo il messaggio di "Nessuna immagine memorizzata"

```
If My.Computer.Clipboard.ContainsImage() = True
                                            Then
```

IL FORMATO DEGLI APPUNTI

I dati presenti negli Appunti possono essere di un formato qualsiasi, definito anche formato degli Appunti. Quando un elemento viene spostato o copiato negli Appunti, gli elementi in altri formati vengono cancellati. Per rendere persistenti gli altri formati, si può utilizzare DataObject, che permette di copiare tutto ciò che esiste negli Appunti correnti, inclusi gli elementi incollati da altre applicazioni. Negli Appunti è possibile inserire una intera classe, a patto che la classe sia serializzabile.

```
picturebox1.Image =
        My.Computer.Clipboard.GetImage()
Else
    MsgBox("Nessuna immagine memorizzata")
End If
```

LAVORARE CON LA RETE

L'Oggetto *My.Computer.Network* fornisce i metodi che permettono l'interazione con la rete cui è connesso il computer.
L'unica proprietà disponibile è la proprietà *My.Computer.Network.IsAvailable*. La proprietà IsAvailable restituisce True se il computer dispone di una rete o di una connessione ad Internet attiva. Quando questa proprietà cambia, l'oggetto scatena un evento *NetworkAvailabilityChanged*. Ad esempio possiamo scrivere il codice seguente che mostra a video lo stato della connessione in rete:

```
If My.Computer.Network.IsAvailable = True Then
        MsgBox("Il Computer è connesso in rete.")
    Else
        MsgBox("Nessuna rete disponibile.")
End If
```

Per determinare la disponibilità di un computer remoto, si può utilizzare il metodo:
My.Computer.Network.Ping
Il metodo Ping accetta un nome di computer, oppure una URL, oppure un indirizzo IP ed un timeout opzionale. Il timeout viene espresso in millisecondi e, se non viene specificato, per impostazione predefinita assume il valore pari a 500 millisecondi. Per verificare se il sito di IoProgrammo è raggiungibile dal nostro PC, possiamo scrivere:

```
If My.Computer.Network.IsAvailable Then
'Quando si specifica un URL,
'non si deve includere http://
        If My.Computer.Network.Ping(
                "www.ioprogrammo.it", 1000) Then
            MsgBox("Indirizzo raggiungibile")
        Else
            MsgBox("Non è possibile raggiungere il
                                        sito.")
        End If
    End If
```

Il metodo *DownloadFile*, può essere utilizzato per eseguire il download di un file, oppure per leggere il contenuto di una pagina HTML identificata dall'indirizzo URL. Questo metodo accetta come parametri opzionali il nome utente e la password, per questo motivo può essere utilizzato anche con siti in cui è necessaria l'autenticazione. È possibile, inoltre, mostrare una finestra di dialogo che mostri lo stato di avanzamento del download.

```
    Try
My.Computer.Network.DownloadFile
                ("http://www.ioprogrammo.it", _
            "c:\Home Page IoProgrammo.html",
                Nothing, Nothing, True, 2000, False)
    Catch ex As Exception
        MessageBox.Show("Il sito di destinazione
                            non è raggiungibile")
    End Try
```

Il metodo UploadFile è simile al metodo DownloadFile e accetta lo stesso insieme di argomenti, fatta eccezione per il secondo parametro, che rappresenta il percorso del file che deve essere caricato in upload all'URL indicato come primo parametro.

GESTIRE IL FILE SYSTEM

L'oggetto My.Computer.FileSystem fornisce molti metodi per l'utilizzo di unità, file e directory. Si possono, ad esempio: copiare, eliminare, spostare e rinominare sia singoli file sia intere directory. È possibile enumerare le cartelle ed i file di una directory, e persino leggere o scrivere un intero file di testo binario in una singola operazione. I metodi sono talmente tanti che non basterebbe l'intero articolo per analizzarli tutti, pertanto ci limiteremo ad osservarne alcuni.

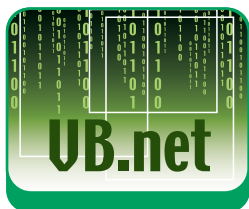
Per analizzare percorsi di file si possono utilizzare i seguenti metodi:

- Il metodo *My.Computer.FileSystem.CombinePath* richiede due percorsi e restituisce un percorso combinato opportunamente formattato.
- ∑Il metodo *My.Computer.FileSystem.GetParentPath* restituisce il percorso assoluto della directory padre del percorso indicato.
- ∑Il metodo *My.Computer.FileSystem.GetFileInfo* restituisce un oggetto FileInfo di cui è possibile determinare le proprietà del file, ad esempio il nome e il percorso.

Se vogliamo determinare il nome ed il percorso di un file restituito dal metodo GetFileInfo, possiamo utilizzare le proprietà Directory Name e Name dell'oggetto FileInfo restituito:

```
Dim FileDiTest As System.IO.FileInfo
```

IL METODO PING

Il metodo Ping non è infallibile nel determinare la disponibilità di un computer remoto, Può portare a risultati errati poiché la porta del ping sulla macchina di destinazione può essere spenta oppure la richiesta di ping può essere bloccata da un firewall o da un router.

```
FileDiTest =
My.Computer.FileSystem.GetFileInfo("C:\MiaCartella\
                                        MioFile.txt")
Dim Cartella As String =
                              FileDiTest.DirectoryName
'mostra l'intero percorso C:\MiaCartella
MessageBox.Show(Cartella)
Dim NomeFile As String = FileDiTest.Name
'mostra MioFile.txt
MessageBox.Show(NomeFile)
```

Alcuni metodi hanno la caratteristica di visualizzare una finestra di dialogo Windows standard, quando eseguono un'operazione su file da cui è possibile rilevare se l'utente clicca sul pulsante Cancel per interrompere l'azione. Se, ad esempio, vogliamo copiare un'intera directory, possiamo utilizzare il metodo CopyDirectory, scrivendo il seguente codice:

```
Try
My.Computer.FileSystem.CopyDirectory
                                ("C:\IoProgrammo", _
        "C:\Backup", FileIO.UIOption.AllDialogs,
                FileIO.UICancelOption.ThrowException)
Catch ex As Exception
        MessageBox.Show("Operazione cancellata
                                          dall'utente")
End Try
```

Quando si avvia l'operazione, viene mostrata la finestra in figura

I metodi *CopyFile*, *MoveDirectory*, *MoveFile*, *DeleteDirectory* e *DeleteFile* presentano le stesse caratteristiche di *CopyDirectory*. Gli ultimi due metodi elencati hanno una ulteriore opzione che permette di inviare i file e le directory eliminate nel cestino di sistema oppure di eliminarli definitivamente senza passare dal cestino

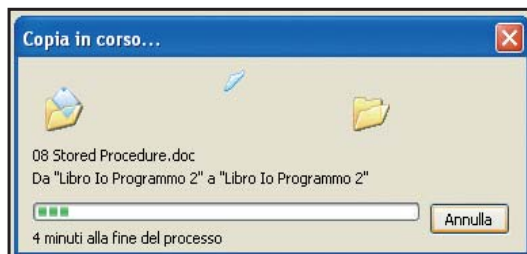



Fig. 2: *Una finestra informativa per la copia dei File*

INFORMAZIONI SUL COMPUTER

Gli oggetti di *My.Computer*: *Clock, Info, Keyboard, Mouse, Ports e Screen* espongono, in buona sostanza, una serie di proprietà a sola lettura che permettono di recuperare le informazioni sul sistema. Il codice seguente mostra in un *TextBox* una serie di informazioni ricavabili da questi oggetti:

```
Dim Informazioni As String = ""
Informazioni = Informazioni & "Data ed Ora Locale "
            & My.Computer.Clock.LocalTime & vbCrLf
Informazioni = Informazioni & "Memoria Fisica " &
  My.Computer.Info.AvailablePhysicalMemory & vbCrLf
Informazioni = Informazioni & "Memoria Virtuale " &
   My.Computer.Info.AvailableVirtualMemory & vbCrLf
Informazioni = Informazioni & "Sistema operativo " &
             My.Computer.Info.OSFullName & vbCrLf

    If My.Computer.Keyboard.CapsLock Then
        Informazioni = Informazioni & "il tasto BLOC
                         MAIUSC è attivo" & vbCrLf
    Else
        Informazioni = Informazioni & "il tasto BLOC
                     MAIUSC NON è attivo" & vbCrLf
    End If

    If My.Computer.Mouse.WheelExists Then
        Informazioni = Informazioni & "il Mouse
          presenta la rotella di scorrimento" & vbCrLf
    Else
        Informazioni = Informazioni & "il Mouse non
          presenta la rotella di scorrimento" & vbCrLf
    End If
Informazioni = Informazioni & "Lo schermo presenta
                      i seguenti Bits Per Pixel " &
           My.Computer.Screen.BitsPerPixel & vbCrLf
Informazioni = Informazioni & "Risoluzione
  Orizzontale" & My.Computer.Screen.Bounds.Width &
                                            vbCrLf
Informazioni = Informazioni & "Risoluzione Verticale"
```

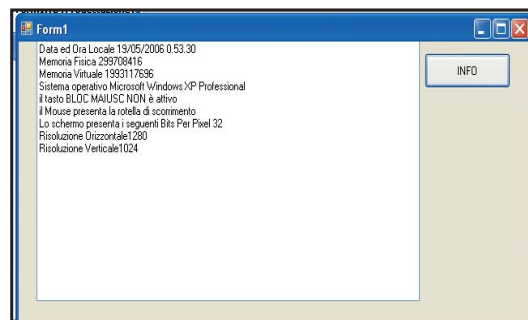



Fig. 3: *Ecco le informazioni sul sistema*

```
        & My.Computer.Screen.Bounds.Height & vbCrLf
TextBox1.Text = Informazioni
```

La maggior parte delle informazioni esposte da questi oggetti sono disponibili utilizzando altri tipi del .NET Framework, ma l'oggetto My.Computer offre una soluzione più semplice da utilizzare.

Luigi Buono

APPLICAZIONI DESKTOP IN JAVA 6

VEDIAMO, ATTRAVERSO ESEMPI UTILI, LE NUOVE FUNZIONALITÀ INTRODOTTE SULLO SVILUPPO DESKTOP DA JAVA 6 MUSTANG, CHE AVVICINANO DI PIÙ JAVA AL SISTEMA OPERATIVO PUR MANTENENDO SEMPRE LA SUA CARATTERISTICA DI PORTABILITÀ

Nel mese scorso è stata rilasciata da SUN la prima Beta ufficiale di Java 6, nome in codice "Mustang". La versione "ufficiale" è prevista per la fine del 2006. Nonostante si tratti di una Beta, si intravede già la direzione che caratterizzerà il linguaggio di SUN nei prossimi anni. Le novità introdotte sono molte e coinvolgono diversi aspetti. In quest'articolo parleremo principalmente delle novità riguardanti le interfacce utente (GUI) che possono essere realizzate con Java.

SVILUPPO DI UN GESTIONALE

L'applicazione che svilupperemo e che ci farà da guida per la scoperta delle nuove funzionalità desktop di Java 6 sarà un gestionale. Come in ogni gestionale avremo bisogno di gestire dati con tabelle, di aprire file e mandarli in stampa e di gestire scelte dell'utente. Chiaramente non analizzeremo lo sviluppo di tutto il gestionale, che alla fine è una semplice applicazione Swing, ma ci soffermeremo sui punti che riguardano le novità introdotte da Mustang in ambito desktop, ovvero

- Splashscreen: È possibile, a livello nativo, avere uno Splashscreen dell'applicazione, ovvero un'immagine visualizzata all'avvio, prima del completo caricamento del programma
- Migliore gestione dell'elemento JTable: ordinamento e filtraggio dei valori
- Tray icon: finalmente è possibile vedere la propria applicazione Java nella barra delle applicazioni in esecuzione, senza dover ricorrere a JNI (Java Native Interface)
- Desktop: Finalmente è possibile da Java avviare i programmi di default associati ai tipi fi file. Allo stesso modo è possibile aprire URL e client email di default del sistema

SPLASHSCREEN

Prima di Java 6, per poter visualizzare uno SplashScreen alla partenza del programma, avevamo bisogno di visualizzare un JFrame per pochi secondi. Un esempio di codice è il seguente:

```
JFrame frame = new JFrame("Vecchio Splash");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_C
                                          OSE);
JLabel label = new JLabel("Vecchio Splash",
                                  JLabel.CENTER);
frame.getContentPane().add(label,
                            BorderLayout.CENTER);
frame.setSize(300, 100);
frame.setVisible(true);
Thread.currentThread().sleep(5000);
frame.getContentPane().remove(frame);
frame.dispose();
System.out.println("Ora parte il programma...");
```

Chiaramente il look di questa finestra iniziale lascia molto a desiderare, pur sforzandosi di ottenere risultati apprezzabili, era difficile ottenere una qualità vicina a quella degli splashscreen tradizionali. Con Java 6 la soluzione è molto più semplice ed elegante. Dal prompt dobbiamo semplicemente avviare l'applicazione in questo modo

```
java -splash:test.png Test
```

Per visualizzare correttamente lo splashscreen possiamo semplicemente sospendere il Thread principale dell'applicazione per 5 secondi all'avvio. In questo modo verrà mostrato lo splashscreen, poi appena il Thread riprenderà l'esecuzione, all'inseri-

mento del primo oggetto grafico, lo splash-screen scomparirà.

```
Thread.currentThread().sleep(5000);
System.out.println("Ora parte il programma...");
JFrame frame = new Jframe("Un vero
                                    SplashScreen");
frame.setDefaultCloseOperation
                              (JFrame.EXIT_ON_COSE);
JLabel label = new JLabel("Un vero SplashScreen",
                                    JLabel.CENTER);
frame.getContentPane().add(label,
                              BorderLayout.CENTER);
frame.setSize(300, 100);
frame.setVisible(true);
```

Fig. 1: **Un esempio di SplashScreen**

L'immagine può essere definita da linea di comando o attraverso il manifest dell'applicazione. Chiaramente nel secondo caso è possibile includere l'intera applicazione (immagine compresa) in un file jar. L'attributo che deve essere usato nel file manifest è SplashScreen-Image. Qui di seguito mostriamo l'esempio di un manifest nel quale viene incluso questo attributo per poter visualizzare l'immagine come splashscreen.

```
Manifest-Version:1.0
Class-Path: lib/libreria.jar
Main-Class: com.javastaff.TestSplashScreen
SplashScreen-Image: test.gif
```

Oltre a questi due metodi per la gestione degli SplashScreen, Java 6 fornisce anche una classe apposita all'interno del package java.awt. Questa classe espone una serie di metodi per poter manipolare l'oggetto grafico Splash Screen tramite Graphics2D, che consente di gestire grafica 2D in maniera semplice.

LA GESTIONE DELLE TABELLE

Il programma d'esempio che stiamo usando come laboratorio per i nostri esperimenti deve ricevere dei dati in input, elaborarli e presentare un output adeguato all'utente. In questo, come in tanti altri programmi gestionali, il tutto passa attraverso la visualizzazione dei dati attraverso una *JTable*. Questa è una classe grafica che dalle prime versioni di Swing in poi è collocata nei package di Java Standard. Permette di gestire un oggetto grafico che rappresenta una tabella classica come quelle di Excel o altri programmi simili. In Mustang sono state introdotte nuove feature riguardanti questa classe. Prima di Java 6 era possibile effettuare l'ordinamento di una *JTable*, ovvero ordinare i dati della tabella in base al valore crescente o decrescente di un campo. Per fare questo però veniva richiesto un bel po' di codice. Mustang introduce un nuovo modo di realizzare l'ordinamento che semplifica molto il lavoro. Vediamone un esempio nel seguente codice

```
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
public class EsempioOrdinamento {
 public static void main(String args[]) {
            String colonne[] = {"Sito", "URL"};
            String righe[][] =  {{"JavaStaff.com",
                          "http://www.javastaff.com"},
              {"News Google", "http://news.google.it"},
              {"JavaLobby",
                         "http://www.javalobby.com"},
              {"HostingJava",
                        "http://www.hostingjava.it"},
              {"IBM Redbooks",
                    "http://www.redbooks.ibm.com"},
              {"JavaCrawler",
                       "http://www.javacrawler.com"},
            };
            TableModel tableModel =
              new DefaultTableModel(righe, colonne) {
             public Class getColumnClass(int column) {
              Class returnValue;
              if ((column >= 0) && (column <
                                getColumnCount())) {
                returnValue = getValueAt(0,
                                  column).getClass();
              } else {
                returnValue = Object.class;
              }
              return returnValue;
             }
            };
            JTable table = new JTable(tableModel);
            RowSorter<TableModel> rowSorter = new
              TableRowSorter<TableModel>(tableModel);
            table.setRowSorter(rowSorter);
            JFrame frame = new JFrame("JTable
```

```
                                            esempio");
frame.setDefaultCloseOperation
                          (JFrame.EXIT_ON_CLOSE);
        JScrollPane pane = new JScrollPane(table);
        frame.add(pane, BorderLayout.CENTER);
        frame.setSize(350, 200);
        frame.setVisible(true);
  }
}
```

Analizziamo ora il codice per capire cosa succede. Prima di tutto definiamo due array di stringhe che rappresentano le righe e le colonne della *JTable*. Passiamo quindi alla definizione di un TableModel. Questa interfaccia ci permette di definire i metodi che la JTable utilizzerà. La parte che ci permette di definire l'ordinamento viene implementata con la definizione di *RowSorter*, al quale passiamo il modello dei dati.
L'ordinamento viene effettuato sulla base del metodo che abbiamo ridefinito (*getColumn Class*). Come potete vedere il modello dei dati viene utilizzato sia da *JTable* che da *RowSorter*. Questo è solo uno dei modi per poter ordinare gli elementi in una *JTable*, a mio avviso quello più semplice.

JTable esempio

Sito ▾	URL
News Google	http://news.google.it
JavaStaff.com	http://www.javastaff.com
JavaLobby	http://www.javalobby.com
JavaCrawler	http://www.javacrawler.com
IBM Redbooks	http://www.redbooks.ibm.com
HostingJava	http://www.hostingjava.it

Fig. 2: **Ordinamento di una JTable**

ALTRI METODI DI ORDINAMENTO>

Le API che sono state aggiunte in Mustang forniscono altre classi e interfacce per manipolare questo tipo di dati e gli eventi associati. Per un maggior approfondimento su questo tema vi rimando alla documentazione ufficiale del package javax.swing.table e delle altre classi correlate (http://java.sun.com/javase/6/docs/api).

FILTRI SUI CONTENUTI

Supponiamo di avere una lunga lista di oggetti e volere visualizzare soltanto quelli che soddisfano una certa regola. Nell'esempio precedente mostravamo una lista di siti con i relativi url, Vediamo come creare un filtro che ci mostri soltanto quelli che contengono una certa stringa all'interno del nome. Fobbiamo definire un JTextField per l'input dell'utente e un listener che filtra la tabella ogni volta che viene cambiato il testo dall'utente. Il seguente esempio di codice mostra come fare

```
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.text.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.awt.*;
public class EsempioFiltraggio {
    public static void main(String args[]) {
        String colonne[] = {"Sito",
                                "URL"};
        String righe[][] =
    {{"JavaStaff.com", "http://www.javastaff.com"},
          {"News Google",
                  "http://news.google.it"},
          {"JavaLobby",
               "http://www.javalobby.com"},
          {"HostingJava",
               "http://www.hostingjava.it"},
          {"IBM Redbooks",
            "http://www.redbooks.ibm.com"},
          {"JavaCrawler",
              "http://www.javacrawler.com"},
        };
        JFrame frame = new
                  JFrame("JTable esempio");
frame.setDefaultCloseOperation
                  (JFrame.EXIT_ON_CLOSE);
        TableModel tableModel =
          new DefaultTableModel(righe,
                              colonne) {
         public Class getColumnClass(in
                              column) {
          Class returnValue;
          if ((column >= 0) && (column
               < getColumnCount())) {
           returnValue = getValueAt(0,
                     column).getClass();
          } else {
           returnValue = Object.class;
          }
          return returnValue;
         }
        };
        JTable table = new
                     JTable(tableModel);
        final TableRowSorter<TableModel>
                         rowSorter = new
    TableRowSorter<TableModel>(tableModel);
        table.setRowSorter(rowSorter);
        final JTextField testoFiltro=new
                         JTextField("");
        testoFiltro.addActionListener(new
                       ActionListener() {
             public void
  actionPerformed(ActionEvent event) {
                         String
```

```
                                        testo=testoFiltro.getText();
                                                          if
                                                   (testo.equals(""))
rowSorter.setRowFilter(null);
                                                          else {
                                                                  try{
rowSorter.setRowFilter(RowFilter.regexFilter(testo));
                                                                  }
catch(Exception e) {
System.out.println(e.toString());
                                                                  }
                                                          }
                                                }
                                        });
                                        JScrollPane pane = new
                                                      JScrollPane(table);
                                        frame.add(testoFiltro,
                                                  BorderLayout.NORTH);
                                        frame.add(pane,
                                                  BorderLayout.CENTER);
                                        frame.setSize(350, 200);
                                        frame.setVisible(true);
                    }
}
```

In questo caso utilizziamo un TableRowSorter al posto del RowSorter che avevamo utilizzato nell'esempio precedente. Abbiamo aggiunto un JTextField per avere un campo di input. A questo componente è associato un ActionListener che viene attivato ogni volta scriviamo qualcosa e mandiamo a capo. In questo caso viene preso il contenuto del JTextField e viene richiamato il metodo setRowFilter. Così facendo il contenuto viene filtrato e la nostra tabella visualizza soltanto le righe che hanno il testo inserito nel JTextField.

JTable esempio

Java

Sito	URL
JavaStaff.com	http://www.javastaff.com
JavaLobby	http://www.javalobby.com
HostingJava	http://www.hostingjava.it
JavaCrawler	http://www.javacrawler.com

Fig. 3: Una jtable disordinata

SYSTEM TRAY & TRAY ICON

Nella maggior parte delle applicazioni Desktop abbiamo la possibilità di avere un menu nella System Tray (ovvero nella barra in basso a destra). Precedentemente a Mustang era possibile utilizzare la System Tray, ma solo attraverso delle librerie opensource. Ora invece è possibile avere queste funzionalità con la distribuzione standard di Java. Chiaramente non è una delle più importanti opzioni che deve avere un programma, ma comunque fa parte di quelle qualità di usabilità che devono far parte di un software Desktop al giorno d'oggi. Senza utilizzare la System Tray potevamo semplicemente minimizzare la finestra che stavamo utilizzando, che però rimaneva comunque nella barra della applicazioni. Per fare ciò si poteva semplicemente cambiare lo stato del JFrame e farlo minimizzare come tutte le applicazioni grafiche, come viene illustrato nel seguente esempio

```
import java.awt.BorderLayout;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class MinimizzaFrameOld extends JFrame
                                   implements ActionListener {
            private JPanel pannello;
            private JButton bottone1;
            private JButton bottone2;
            public MinimizzaFrameOld() {
                        try {
                                    pannello=new JPanel();
                                    pannello.setLayout(new
                                           BorderLayout());
                                    bottone1=new
                                                  JButton();
bottone1.setText("Minimizza");
bottone1.addActionListener(this);
                                    bottone2=new
                                                  JButton();
bottone2.setText("Invisibile");
bottone2.addActionListener(this);
pannello.add(bottone1,BorderLayout.NORTH);
pannello.add(bottone2,BorderLayout.SOUTH);
this.getContentPane().add(pannello,
                                     BorderLayout.CENTER);
                                    this.setSize(200,100);
                                    this.setVisible(true);
                        }
                        catch(Exception e) {
System.out.println(e.toString());
                        }
            }
            public void actionPerformed(ActionEvent
                                                event) {
                        if(event.getSource() ==
                                             bottone1) {
this.setExtendedState(this.ICONIFIED);
```

DOVE POSSO SCARICARE JAVA 6?

La beta di Java 6 è scaricabile dall'indirizzo http://java.sun.com/javase/6/download.jsp

```
        }
        else if(event.getSource() ==
                              bottone2) {
            this.setVisible(false);
        }
    }
    public static void main(String[] ar) {
        MinimizzaFrameOld mf = new
                        MinimizzaFrameOld();
    }
}
```

L'esempio riportato crea un semplice *JFrame* e aggiunge due *JButton*. Su tutti e due i bottoni vengono aggiunti degli *ActionListener*, per sapere quando sono stati premuti. Premendo il primo bottone facciamo semplicemente abbassare la nostra finestra, settando lo stato del nostro *JFrame* a *ICONIFIED*. Il secondo bottone invece rende invisibile il nostro frame, utilizzando il metodo *setVisible*(). Così facendo non possiamo comunque permettere all'utente di riaprire l'applicazione, visto che non può più vedere il frame principale. Utilizzando invece la System Tray possiamo decidere che quando l'utente vuole minimizzare la nostra applicazione, questa scompare e successivamente può essere riportato in primo piano semplicemente cliccando sull'icona. Prima di tutto dobbiamo controllare se la versione che stiamo utilizzando supporta la System Tray

```
if (SystemTray.isSupported())
        aggiungiSystemTray(mf);
```

Passiamo quindi alla definizione della System Tray. Dobbiamo catturare l'evento clic del mouse e soprattutto dobbiamo render disponibile un comando per uscire dall'applicazione, perchè anche quando clicchiamo sulla X della finestra in realtà rendiamo invisibile il frame, ma non chiudiamo l'applicazione.

```
public static void
        aggiungiSystemTray(MinimizzaFrame mf) {
        final TrayIcon trayIcon;
        final MinimizzaFrame mfm=mf;
        SystemTray tray =
                SystemTray.getSystemTray();
        Image image =
    Toolkit.getDefaultToolkit().getImage("tray.gif");

        MouseListener mouseListener =
                    new MouseListener() {

            public void
                mouseClicked(MouseEvent e) {
System.out.println("Evento del mouse");
                    mfm.setVisible(true);
            }
            public void
                mouseEntered(MouseEvent e) {
            }

            public void
                mouseExited(MouseEvent e) {
            }

            public void
                mousePressed(MouseEvent e) {
            }

            public void
                mouseReleased(MouseEvent e) {
            }

        };

        ActionListener listener = new
                    ActionListener() {
            public void
                actionPerformed(ActionEvent e) {
                System.out.println("Uscita dal
                            programma");
                System.exit(0);
            }
    };

    PopupMenu popup = new PopupMenu();
    MenuItem defaultItem = new
                    MenuItem("Esci");
    defaultItem.addActionListener(listener);
    popup.add(defaultItem);

    trayIcon = new TrayIcon(image,
            "MinimizzaFrame", popup);

    trayIcon.setImageAutoSize(true);
    trayIcon.addMouseListener(mouseListener);

    try {
        tray.add(trayIcon);
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}
```

Analizziamo ora cosa succede. Prima di tutto viene istanziato un oggetto *SystemTray* utilizzando il metodo statico *getSystemTray*() della stessa classe. Questo oggetto rappresenta la SystemTray, dove noi andiamo ad aggiungere la nostra icona, con il relativo menu. La classe che ci permette di aggiungere l'icona è TrayIcon, che inizializziamo con un'immagi-

ne, una stringa ed un Menu. Questo menu è il classico *PopupMenu*, dove possiamo aggiungere semplici elementi *MenuItem* oppure cose più elaborate tipiche di Swing. In questo caso viene aggiunto un semplice elemento, che ci permetterà di uscire dall'applicazione. Per capire che l'icona è stata cliccata dobbiamo semplicemente aggiungere alla nostra *TrayIcon* un *MouseListener*, nel quale implementiamo soltanto il metodo *mouseClicked*. *MouseListener* è un'interfaccia che ci notifica gli eventi relativi al mouse. In questo metodo settiamo semplicemente a true la visibilità del frame, permettendo quindi alla nostra applicazione di sparire e riapparire dal desktop.

Fig. 4: Icona dell'applicazione nella barra di sistema

DESKTOP API

Veniamo ora alla parte più interessante (almeno a mio avviso) della piattaforma Mustang lato Desktop. Abbiamo già visto che le precedenti feature che sono state introdotte cercano di dare sempre più potere alle applicazioni Java, rendendole sempre più integrate nel sistema operativo dove devono girare. Uno dei problemi storici delle applicazioni Java Desktop è stato appunto quello di avere poteri limitati rispetto a programmi scritti con linguaggi di programmazione nativi come C e C++. Un altro grande problema è sempre stato il Look & Feel delle applicazioni, che si discostava molto da quello classico di tutte le applicazioni grafiche. Con Mustang si è voluto appunto riavvicinare il mondo di Java al Desktop e una delle API più interessanti in questo senso è la Desktop API. Questa permette allo sviluppatore Java di interagire con il sistema operativo, in un modo sicuramente maggiore rispetto a quello che veniva precedentemente offerto. Questa API deriva dal progetto opensource JDIC (JDesktop Integration Components), che è stato inglobato all'interno dell'ultima release di casa SUN. Quello che ora vedremo è la classe Desktop, che ci permette di

- Avviare il browser predefinito
- Avviare il client di posta predefinito
- Aprire un file con l'applicazione ad esso associata
- Modificare un file con l'applicazione ad esso associata
- Stampare un file

Incominciamo a vedere un po' di codice, riguardante le prime due feature, ovvero come poter avviare browser e client di posta.

```
import java.awt.Desktop;
import java.io.*;
import java.net.*;

public class EsempioBrowserMail {

        private static Desktop desktop;

        public static void main(String a[]) {
                if
                (Desktop.isDesktopSupported()) {
                        desktop =
                        Desktop.getDesktop();
                }
                else {
System.out.println("In questo sistema non è
                abilitato il supporto Desktop");
System.exit(0);
                }
System.out.println("Apriamo il browser");
lanciaBrowser("www.javastaff.com");

System.out.println("Apriamo il client email");
lanciaMailClient("doc@javastaff.com");

        }
```

In questo primo pezzo di codice effettuiamo il controllo relativo al supporto alla JDesktop API. Qesto controllo deve essere effettuato perché non tutti i sistemi operativi potrebbero supportarlo. Se la feature che vogliamo utilizzare è supportata allora passiamo direttamente all'esecuzione dei due metodi che lanciano il browser e il client email.

```
public static void lanciaBrowser(String testo) {
                URI uri = null;
                try {
                uri = new URI(testo);
                desktop.browse(uri);
                }
                catch(Exception e) {
        System.out.println(e.toString());
```

LA SYSTEM TRAY IN JAVA 5

Prima di Mustang era già possibile utilizzare la System Tray in Java. Sono infatti presenti nel panorama opensource diverse librerie che permettono di interagire con la System Tray senza utilizzare l'ultima versione di Java
- **Jan Struyf's Windows Tray Icon http://jeans.studentenweb.org/java/trayico/trayicon.html**
- **systray4j http://systray.sourceforge.net**
- **JDIC https://jdic.dev.java.net**
- **Eclipse SWT Tray http://wiki.eclipse.org/index.php/Development_Resources**

```
        }
}
```

In questo metodo non facciamo altro che inizializzare un oggetto URI, passando come parametro la stringa dell'argomento. Se quello che abbiamo passato non è un URI (Uniform Resource Identifier) allora verrà lanciata un'eccezione. Se invece riusciamo ad inizializzare l'oggetto dobbiamo solo richiamare il metodo browse di Desktop e il browser predefinito si avvierà cercando di aprire quello che abbiamo richiesto. Passiamo ora al client email

```
public static void lanciaMailClient(String testo) {
        URI uri = null;
        try {
        if (testo.length() > 0) {
                uri = new URI("mailto",
                        testo, null);
                desktop.mail(uri);
        } else {
                desktop.mail();
        }
        }
        catch(Exception e) {
        System.out.println(e.toString());
        }
}

}
```

L'AUTORE

L'autore, Federico Paparoni, può essere contattato per suggerimenti o delucidazioni all'indirizzo email federico.paparoni@javastaff.com

Anche in questo metodo creiamo un oggetto URI, al quale passiamo l'email che abbiamo passato come parametro. Successivamente richiamiamo il metodo *mail*() di Desktop che avvia il client di posta predefinito. Vediamo infine come poter aprire, modifica e stampare un file utilizzando sempre la Desktop API

```
Desktop desktop;
String filename = "c:\\prova.txt";
File f = new File(filename);
if (Desktop.isDesktopSupported()) {
        desktop = Desktop.getDesktop();
}
else {
        System.exit();
}
try {
        //APRIAMO IL FILE
        desktop.open(file);
        //MODIFICHIAMO IL FILE
        desktop.edit(file);
        //STAMPIAMO IL FILE
        desktop.print(file);
}
catch(Exception e) {
        System.out.println(e.toString());
}
```

CONCLUSIONI

Abbiamo visto che le nuove funzionalità di Mustang permettono di avere delle applicazioni grafiche migliori rispetto alle vecchie versioni di Java. C'è da dire che anche le prestazioni di questa versione sembrano essere migliorate rispetto alle precedenti. Le novità introdotte nella grafica non sono solo quelle che abbiamo visto, infatti molte riguardano anche lo sviluppo enterprise. Come sempre il passaggio ad una nuova versione ricca di novtià così importanti richiederà un po' di tempo, tuttavia siamo certi che le innovazioni apportate favoriranno una transizione rapida.

Federico Paparoni

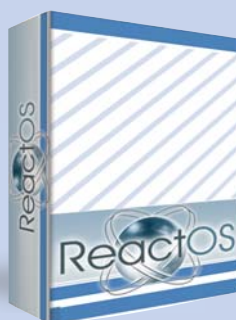
IL PROGETTO JDESKTOP INTEGRATION COMPONENTS

Molte delle attese che si sono formate intorno al nascente Java 6 ruotano al progetto JDIC, ovvero JDesktop Integration Components. Come avrete abbondantemente appreso da questo articolo JDIC si propone una maggiore integrazione fra Java e i sistemi su cui i programmi Java devono ruotare. L'intero progetto nasce sul sito javadesktop.org, e da qui si possono reperire informazioni complete su tutte le funzionalità che ne caratterizzeranno l'uso in Mustang, ovvero Java 6. Le motivazioni che hanno indotto Sun a scegliere questa direzione sono molteplici. Prima di tutto si è tenuto conto della crescente diffusione di Linux sul lato Desktop. In secondo luogo ci si è resi conto della comodità di poter utilizzare un'interfaccia Standalone rispetto ad un'interfaccia Web. Infine la tecnologia "Java Web Start" sembra avere riscosso un certo successo. Tutti motivi per cui, la scelta di favorire lo sviluppo di applicazioni maggiormente integrate con i Desktop in cui vivono sembra essere vincente.D'altra parte Java ha alcune cose in comune con HTML e con le applicazioni che girano in un Browser, ovvero la possibilità di funzionare allo stesso modo su qualunque piattaforma. Rispetto alle web application ha dalla sua parte la possibilità di girare con delle interfacce standalone. Purtroppo fino a ieri queste enormi possibilità non erano supportate da una serie di API che potessero sfruttare a fondo il sistema operativo ospite. Con Mustang e JIC questa lacuna viene finalmente colmata. Il lavoro è stto duro, in quanto si doveva fare in modo di garantire lo stesso look & feel adottato dal sistema, con l'ambiente Java, ma alla fine il risultato è soddisfacente. Java 6 sarà in grado di utilizzare menu, popup, bottoni, barre, e colori del sistema operativo su cui le applicazioni girano.

SOFTWARE SUL CD

ReactOS 0.3.0 Rc1

IL PRIMO CLONE OPENSOURCE DI MICROSOFT WINDOWS

ReactOS è un progetto che da tempo fa parlare di se. L'intenzione era ed è quella di creare un sistema operativo completamente OpenSource con compatibilità binaria con gli eseguibili di Windows. Recentemente è stata rilasciata la Release Candidate 1 della versione 3 che ha raggiunto una maturità tale da portare il progetto, finalmente, su un piano pratico. In realtà lo scopo degli sviluppatori di ReactOS non è semplicemente quello di emulare il più noto fratello maggiore e di raggiungere la compatibilità binaria con gli eseguibili, piuttosto quello di continuare ad innovare, di modo da poter raggiungere lo stato dell'arte della tecnologia con un sistema che diventerebbe non un esatto clone di Windows, ma un suo derivato che ne estenderebbe e ne migliorerebbe la qualità. La sfida è difficile. Molto affidamento viene fatto sulla disponibilità dei sorgenti, grazie ai quali si spera che molti programmatori contribuiranno allo sviluppo con modalità che sono tipiche dei software OpenSource. La sfida è emozionante, tanto che noi stessi in queste pagine vi presentiamo sia una versione live che vi consente di valutare lo stato d'avanzamento del progetto, sia i sorgenti della versione 0.3.0 RC1 di modo che chiunque di voi possa, se vuole, contribuire allo sviluppo

Directory:/ ReactOS

HIBERNATE 3.1.3

METTE D'ACCORDO SQL E PROGRAMMAZIONE OBJECT ORIENTED

Ne parliamo spesso su ioProgrammo, in quanto si tratta di uno dei problemi che sta maggiormente influenzando il modo di programmare nell'ultimo anno. Come far si che la logica del linguaggio SQL sia integrata nel modo di pensare ad oggetti tipico di un programmatore?
Il tool che più di ogni altro ha fatto passi da gigante in questo settore è Hibernate, che grazie a semplici file XML riesce a trasformare, tabelle, righe, colonne, celle di un database nelle corrispondenti Classi di codice. In questo numero presentiamo un primo articolo scritto dal bravo Fabrizio Fortino che ci illustra appunto come questo framework può migliorare la qualità della programmazione. Nell'articolo viene anche presentato un plugin per Eclipse, che come sempre non manca di estensioni che facilitano lo sviluppo.

Directory:/ Hibernate

SKYPE CONTACTS

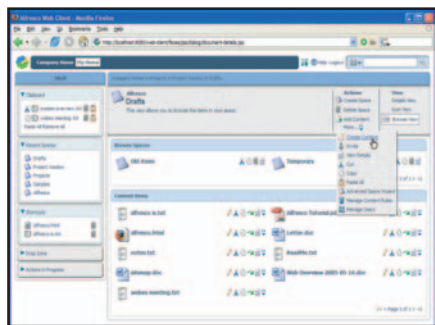
LE API PER SVILUPPARE ESTENSIONI PER SKYPE

Skype è stato il primo software a mostrare la potenza del voice over ip. Attualmente è usato da milioni di utenti in Italia e nel mondo. In questo numero di ioProgrammo vi presentiamo un bell'articolo del nostro Paolo Perrotta che mostra come scrivere un bot autorisponditore per Skype. Strumento essenziale per compiere questa operazione, oltre allo stesso Skype anche una serie di oggetti COM che fanno da interfaccia fra il nostro codice e Skype stesso. La tecnica è affascinante e se avrete modo di tentare un approccio vi accorgerete di quanto siano ben strutturate queste librerie.

Directory:/ SkypeContacts

ALFRESCO 1.2.1

IL CONTENT MANAGEMENT SECONDO JSP

Semplicità è la parola d'ordine per accedere alla logica operativa di Alfresco, un innovativo sistema di content management che sta riscontrando un certo successo presso gli sviluppatori di tutto il mondo. Padre di Alfresco è John Newton, un nome che tra i "consumatori" di CMS dovrebbe suscitare una certa reazione, poiché stiamo parlando del cofondatore di Documentum, tra i primissimi sistemi di content management che già negli anni '90 spopolava presso la nicchia degli addetti ai lavori. A distanza di tre lustri, le cose sono cambiate anche nel mondo CMS, dove, a fronte dello sviluppo senza limiti di Internet, coinciso con l'esplosione

della filosofia open source, e, con esso, del consolidamento del concetto di portale come spazio virtuale in cui ospitare contenuti dalla natura dinamica, si è assistito alla proliferazione di strumenti in grado di gestire la pubblicazione di informazioni senza presupporre la conoscenza di linguaggi di programmazione semplici o complessi. Oggi bastano pochi clic per allestire una struttura efficiente da mettere in linea, e trasformare il proprio spazio Web in realtà viva e pulsante, alla cui vivacità poter contribuire in maniera diretta ed immediata: in quest'ottica, Alfresco si pone l'obiettivo, ambizioso ma non impossibile, di rappresentare un punto fermo nel panorama CMS internazionale. Tra l'altro, Alfresco si offre come sistema di sviluppo e pubblicazione di contenuti su Web sfruttando le Java Server Faces, l'innovativa tecnologia che nasce con l'obiettivo di semplificare lo sviluppo di applicazioni Web oriented attraverso il supporto di componenti più ricchi e completi che migliorino l'aspetto e la funzionalità delle interfacce utenti

Directory:/ Al fresco

DOTNETNUKE 4.3.0

UN FRAMEWORK PER LA COSTRUZIONE DI CMS

Era nato come un semplice CMS ma nel tempo si è evoluto in modo rapidissimo ed è diventato un CMS strutturato in modo piramidale. Chi vuole può installarlo come semplice sistema di Document Management, ed in questo senso potrà sfruttare le funzioni esposte e non sono certamente poche. Il layout è molto personalizzabile, i moduli sono molti e facilmente integrati. Coloro i quali invece

necessitano di una maggiore personalizzazione possono pensare di mettere le mani direttamente nei sorgenti. Addirittura esiste uno starter kit per Microsoft Visual Studio 2005. Certamente il codice e complesso e persino l'adozione dello Starter Kit presenta qualche difficoltà iniziale nell'utilizzo, tuttavia superati i primi scogli, si avverte subito come la modularità del codice e la forte propensione alla programmazione Object Oriented siano forniscano una base di sviluppo sicura e affidabile

Directory:/ DotNetNuke

DRUPAL 4.7.2

LA NUOVA VERSIONE DELLA PIATTAFORMA PER BLOGGERS OPENSOURCE

Un blog è un contenitore singolo di informazioni. Una piattaforma per bloggers è un sistema che include in se più di un blog. Esempi concreti possono essere Splender, o Bloggers. Drupal è una piattaforma per bloggers scritta in PHP estremamente efficiente. Le sue caratteristiche più interessanti sono quella della modularità e della flessibilità del layout. Dal punto di vista del programmatore è utile sapere che Drupal espone un intero framework per la costruzione dei moduli. Se avrete occasione di dare uno sguardo ai sorgenti, e agli esempi vi accorgerete che il framework in questione è concepito in maniera assolutamente solida ed è stato progettato in maniera estremamente efficace. Costruire un modulo per Drupal non richiede sforzi eccessivi, il framework nasconderà i dettagli dell'integrazione con il sistema principale e vi lascerà concentrare solo sulle funzionalità di cui vorrete dotare i vostri moduli

Directory:/ Drupal

YETANOTHER FORUM 1.0.1

ANCORA UN ALTRO FORUM!

Il gioco di parole si presta bene per marcare la vocazione OpenSource di questo progetto. Di fatto, tipicamente questo genere di gioco di parole viene applicato proprio a software OpenSource. YetAnotherForum è un forum scritto interamente in .NET. Il progetto è piuttosto ambizioso e si pone sulla scia di progetti ben più noti quali phpBB. Tuttavia è importante notare oltre alle qualità intrinseche di questo forum che sono veramente eccellenti, quanto anche in ambienti .NET si stia cominciando a

diffondere il concetto di OpenSource. Per quanto riguarda questo particolare forum, siamo convinti che ben presto lo vedremo in versioni riviste e migliorate proprio grazie all'apporto della community. Per adesso non vi resta che provarlo…

Directory:/yetanotherforum

PHPBB

IL FORUM PER ECCELLENZA

Scritto in PHP con l'ausilio di MySQL è stata probabilmente la prima appli-

cazione OpenSource a proporre un modello per lo sviluppo di forum sul Web.

Attualmente è probabilmente il prodotto che espone il maggior numero di funzionalità, più facilmente estendibile e più ricco di moduli. In alcune versioni sono stati riscontrati problemi di sicurezza, tuttavia prontamente risolti. Se avete bisogno di installare un forum o più semplicemente volete studiare alcune tecniche per scrivere codice elegante e funzionale in PHP, probabilmente PHPbb è il software che fa per voi

Directory:/ PHPbb

WIKICALC 0.9.1

IL PRIMO FOGLIO ELETTRONICO COLLABORATIVO

Dalla geniale mente dei creatori di MediaWiki, il software per la scrittura di documentazione collaborativa, arriva questo WikiCalc, la prima web

application che consente di condividere esattamente come se fosse un wiki, formule, dati e fogli elettronici. Si tratta di un modello interessante del quale sicuramente sentiremo ancora parlare nei prossimi anno.

Directory:/ WikiCalc

WORDPRESS

IL PRIMO BLOG!

All'inizio c'era PHPNuke, poi sono arrivati Xoom e gli altri, e il primo a seguire il modello "Blog" che ormai conosciamo così bene è stato probabilmente Wordpress. Si tratta di un sistema molto semplice e veloce da utilizzare, ma allo stesso tempo anche molto completo ed efficace. Se non volete incorrere in inutili complicazioni, è lo strumento che fa per voi.

Directory:/ Wordpress

JOOMLA 1.0.8

L'EREDE DI MAMBO

Nato da una costola del famoso CMS Mambo, Joomla sta diventando rapidamente un punto di riferimento per coloro che hanno bisogno di un sistema potente, stabile, altamente personalizzabile. Scritto in PHP è compatibile con tutta una serie di database. E' facilmente estendibile grazie alla disponibilità di moduli aggiuntivi, ed è programmabile grazie al framework

che fa da base all'intero CMS. Si tratta ormai di un prodotto consolidato a cui ci si può rivolgere senza troppe esitazioni per coprire un elevato numero di esigenze

ECLIPSE SDK 3.1.2

L'AMBIENTE APERTO

Eclipse è un IDE "Aperto" multipiattaforma e completamente scritto in Java. Per aperto si intende un sistema che può essere facilmente espandibile per mezzo di moduli. Perciò se da un lato inizialmente si presenta pronto per favorire lo sviluppo di applicazioni Java, dall'altro lato tramite moduli può diventare un comodo IDE per PHP oppure per C++ oppure per XML, oppure per qualunque altro linguaggio che sia supportato da un modulo. In questo numero di ioProgrammo lo abbiamo utilizzato in più occasioni ed è ormai diventato uno standard per ogni programmatore Java, tuttavia le estensioni di cui dispone incominciano a essere veramente notevoli. Unica nota a suo demerito, una certa pesantezza che caratterizza tutto l'ambiente

Directory:/ EclipseSDK312

JAVA DEVELOPMENT KIT 1.5.0 UPGRADE 7

INDISPENSABILE PER PROGRAMMARE IN JAVA.

Ci corre l'obbligo, prima di tutto, di fare i nostri migliori auguri a java che ha appena compiuto 10 anni di vita. Il linguaggio di Sun nato con l'ambizione di essere il primo realmente multipiattaforma e che portava con se la grande ambizione di servire qualunque tipo di periferica, dopo 10 anni di esistenza può dire di avere largamente realizzato i suoi obiettivi. Con una base di programmatori larghissima e con il primato invidiabile di essere il linguaggio base per i telefonini di nuova generazione come per i PC come per i sistemi embedded è ad oggi uno dei linguaggi più diffusi al mondo. Per programmare in Java è necessario semplicemente dotarsi del J2SE che vi presentiamo in questo stesso numero, tutto il resto sono AddON, anche se non neghiamo che un buon editor aiuta. Non vi resta che installare il JDK dotarvi di entusiasmo e pazienza e creare il vostro primo "Hello Word" carta di ingresso del meraviglioso mondo di Java.

Directory:/ J2SE7

RUBY 1.8.4

IL LINGUAGGIO EMERGENTE

Avevamo cominciato a parlarvene nel numero 104 di ioProgrammo. Questo mese utilizziamo Ruby per sviluppare un bot per skype. Se avrete la pazienza di leggere l'articolo, scoprirete quanto sia facile con Ruby accedere ad un web services, interfacciarsi con oggetti COM, gestire l'intero ciclo dello sviluppo. E' in linguaggio veramente interessante sul quale vi chiediamo di farci conoscere le vostre impressioni e che vi invitiamo a provare anche in caso di progetti in produzione

Directory:/ Ruby

IL DILEMMA DEL PRIGIONIERO

CI SIETE VOI, IL VOSTRO COMPARE E IL BANCO. AVETE A DISPOSIZIONE DUE CARTE: "TRADISCI" E "COLLABORA", I PUNTI LI DA IL BANCO. SE LE CARTE CORRISPONDONO FORSE CE LA FATE, ALRIMENTI... DIAMO UNO SGUARDO AD UN ALGORITMO CLASSICO

E pensare che era quasi fatta... Dopo una dura vita di furti e ruberie, Al e Jack pensavano di chiudere la società con un'ultima rapina in banca. E invece eccoli lì, pizzicati dopo il colpo, interrogati dalla polizia in stanze separate. Ma non tutto è perduto: alla polizia mancano le prove.
Sotto la lampada dell'interrogatorio, i due non perdono il loro sangue freddo, né le loro notevoli capacità logiche. "Potrei tradire il mio socio", riflette Al. "Uscirei subito di galera e mi godrei il malloppo per conto mio mentre lui marcisce dietro le sbarre. Oppure potrei tenere la bocca cucita, scontare una piccola pena per porto d'armi illegale e dividere il grisbì con il mio compare... Sempre che lui non mi tradisca, nel qual caso la mia pena sarebbe durissima e quel disonesto di Jack terrebbe i soldi tutti per sé". Jack, nel frattempo, fa esattamente gli stessi ragionamenti.
Cosa faranno Al e Jack? Sceglieranno di spifferare tutto, o terranno il becco chiuso? Questo dubbio ha appassionato per oltre vent'anni economisti, biologi e matematici.

IL CRIMINE PAGA

Il classico Dilemma del Prigioniero funziona così: ci sono due giocatori e un banco. Ciascun giocatore ha due carte: una carta C ("Collabora") e una carta T ("Tradisci"). Entrambi i giocatori giocano una carta coperta. Il banco scopre le due carte, e in base al risultato assegna un "premio" o una "punizione" a ciascun giocatore. Se entrambi collaborano (giocano C), a tutti e due tocca un modesto "Premio per la Collaborazione". Se entrambi tradiscono (giocano T), a ciascuno andrà una modesta "Punizione per il Tradimento". Se uno dei due gioca T e l'altro gioca C, chi ha tradito prende una corposa sommetta, la "Tentazione per il Tradimento"; chi ha ingenuamente collaborato si becca una dura punizione, che chiameremo la "Stangata dell'Allocco". Non esistono altre possibilità: i giocatori possono giocare solo CC, TT, CT o TC. Entro certi limiti, non è importante a quanto ammontano esattamente i premi e le punizioni, e nemmeno se le punizioni siano delle vere punizioni o semplicemente dei premi di valore più basso. Tutti i premi possono essere espressi in "punti". Possiamo assegnare 0 punti per la Stangata, 1 per la Punizione, 3 per il Premio e 5 per la Tentazione. In figura 1 puoi vedere una semplice tabella dei premi in base alle giocate.
Ora diciamo che voi siete Al. Sapete che Jack è un tipo senza tanti scrupoli, che farà solo il proprio interesse. Anche voi, d'altra parte, non siete uno stinco di santo. Cosa giochereste? Provate a pensarci un po' prima di dare la risposta.
Se io fossi Al, non avrei dubbi. Il mio ragionamento sarebbe questo. Poniamo che Jack giochi T. Allora anche a me converrebbe giocare T. Ci beccheremo entrambi la Punizione per il Tradimento, ma sarebbe sempre meglio della Stangata che prenderei giocando C. E se Jack dovesse generosamente giocare C? Be', mi converrebbe comunque giocare T. Il povero Jack si beccherebbe la Stangata, e io me ne tornerei a casa con la Tentazione per il Tradimento. Se avessi giocato C avrei dovuto accontentarmi del meno consistente Premio per la Collaborazione. Insomma, non si scappa: è triste dirlo, ma comunque la si giri mi converrebbe tradire il mio ex socio. Ecco perché il Dilemma del Prigioniero è considerato un paradosso: non ci sono modi di uscirne puliti. A meno che Al e Jack non si siano messi d'accordo e possano fidarsi ciecamente l'uno dell'altro, lo scenario ideale di una piena collaborazione non è realizzabile in pratica. Qualcuno sarà stangato. Ma è davvero così?

GUERRE FREDDE E UCCELLI PULCIOSI

Il motivo per cui il Dilemma del Prigioniero è così affascinante è che il mondo reale ne è pieno. In particolare, gli economisti trovano Dilemmi come questo ovunque. Se la mia azienda è in concorrenza diretta con un'altra, mi conviene o no fare pubblicità? La pubblicità costa, ma se io faccio pubblicità e il mio con-

corrente no, sarà un bel colpo per noi e una stangata per lui (TC). Se entrambi facciamo pubblicità (TT), spendiamo soldi per lasciare la situazione più o meno com'è. Se nessuno dei due la fa (CC), la situazione resta com'è ma risparmiamo i soldi. Ci converrebbe collaborare entrambi e non fare pubblicità, ma non possiamo esporci al rischio che il concorrente "tradisca".
Il Dilemma salta fuori ovunque anche in politica. Un caso classico è la Guerra Fredda. America e URSS avrebbero potuto diminuire le spese per gli armamenti, ma questo li esponeva al rischio che l'avversario continuasse a studiare armi più sofisticate per l'attacco finale. Sarebbe stato meglio per tutti smettere di costruire armi, eppure la corsa al riarmo continuava. Alla fine uno dei due contendenti è uscito di scena per altri motivi, ed stata è una fortuna: il Dilemma del Prigioniero insegna che ad entrambi sarebbe convenuto attaccare per primi alle prime avvisaglie di "tradimento" dell'avversario.
Un altro campo nel quale il dilemma salta fuori spesso è la biologia. Richard Dawkins (vedi box: "Il gene egoista") descrive una specie di uccelli afflitta da un dilemma morale. Ciascun uccello riesce a spulciarsi col becco in tutto il corpo, tranne che in un punto: la cima della testa. Può farsi spulciare la testa da un amico, e ricambiargli il favore. Spulciare gli altri costa tempo, ma alla fine conviene a tutti. Se però un uccello si trova davanti ad un traditore, allora prenderà una Stangata: perderà tempo per spulciare l'altro, ma alla fine si terrà la testa pulciosa.
La cieca selezione genetica avrebbe dovuto punire duramente l'altruismo in casi come questo. La fredda logica del Dilemma ci dice che a nessuno conviene collaborare. In una popolazione di animali altruisti, un egoista avrebbe vita comodissima: non farebbe alcuna fatica a spulciare gli altri, e sarebbe sempre ben spulciato. Questo animale antipatico sarebbe favorito dalla selezione naturale, e avrebbe più probabilità degli altri di riprodursi anziché morire per qualche malattia portata dalle pulci. Riproducendosi, passerebbe ai suoi figli il "gene dell'egoismo" anziché quello che porta l'istinto di spulciamento reciproco. Nel corso delle generazioni l'egoismo dovrebbe diffondersi tra la popolazione, mentre l'altruismo, svantaggiato, finirebbe per soccombere. La cosa strana, però, è che in natura molte specie di uccelli e altri animali si spulciano generosamente a vicenda. Perché, dunque, questo istinto dell'altruismo ha prosperato anziché estinguersi?
L'egoismo ha un problema. Senza dubbio, un egoista in una popolazione di altruisti fa una gran bella vita. Ma se il nostro egoista diffonde il suo codice genetico, presto non sarà più l'unico egoista della combriccola. Dopo qualche generazione, gli egoisti saranno tanti. E mentre un egoista in una popolazione di altruisti prospera, un egoista in una popolazione di egoisti è nei guai. Il nostro egoista non dovrebbe pagare il prezzo di spulciare gli altri, ma questo gli sarebbe poca consolazione nel momento in cui dovesse morire di pulci lui stesso perché non trova in giro abbastanza altruisti disposti a spulciarlo. Per usare un termine da biologo, l'egoismo nello spulciamento non è una Strategia Evolutivamente Stabile (ESS), nel senso che non è "sostenibile". Gli egoisti vanno alla grande per un po', e poi si mettono nei guai da soli.
Che differenza c'è tra questa situazione e il Dilemma che abbiamo descritto? Una sola: il Dilemma si gioca una volta sola, o la va o la spacca. Le situazioni che si trovano in natura si giocano continuamente. E' come se avessimo tante partite di Dilemma del Prigioniero in sequenza. Questo fa tutta la differenza, perché non posso più sperare che un tradimento passi impunito. Se tradisco in un turno, posso aspettarmi ritorsioni nel turno successivo. Forse, in queste condizioni, l'altruismo paga? Cerchiamo di scoprirlo con l'aiuto di un computer.

CRIMINI E VENDETTE

La variante del Dilemma che ci interessa si chiama Dilemma del Prigioniero iterativo. È come una serie di partite del Dilemma giocate in sequenza. Alla fine, chi ha più punti vince. Tutto qui, a parte poche regole aggiuntive (per conoscerle, leggi il box: "L'ombra del futuro"). Il dilemma iterativo fu proposto nel 19XX, e subito messo alla prova con un concorso tra programmi per computer. Ciascun programma giocava con una strategia diversa, e dovette giocare contro tutti i propri avversari. Alla fine emerse un chiaro vincitore. Il gioco è interessante, e vale la pena di provarlo. Possiamo scrivere un semplice programma Java per giocare delle partite di Dilemma Iterativo. Per prima cosa definiamo le mosse (in Java 5):

```
public enum Mossa {
  C, T;
}
```

Ecco la classe base per tutte le strategie di gioco:

```
public abstract class Prigioniero {
  public int punti = 0;
  public abstract Mossa gioca();
  public void mossaAvversaria(Mossa m) {
  }

  @Override
  public String toString() {
    return getClass().getSimpleName();
  }
}
```

Ciascun Prigioniero ricorda quanti punti ha. Il metodo gioca(), che è astratto e va implementato dalle sottoclassi, restituisce la prossima mossa. Il me-

todo mossaAvversaria() serve al banco per informare il giocatore dell'ultima mossa dell'avversario. Come abbiamo accennato, questa può essere un'informazione importante per un giocatore, ma di default questo metodo non fa niente (il giocatore non ha memoria delle mosse passate). Il metodo toString() restituisce semplicemente il nome della classe. Ora ci serve un programma per gestire l'intera partita:

```
public class DilemmaDelPrigioniero {

  public final static int TENTAZIONE = 5;
  public final static int PREMIO = 3;
  public final static int PUNIZIONE = 1;
  public final static int FREGATURA = 0;
  public final static int TURNI = 100000;
  final Prigioniero al;
  final Prigioniero jack;

  public DilemmaDelPrigioniero(Prigioniero p1,
        Prigioniero p2) {
     al = prigioniero1;
     jack = prigioniero2;
  }
  private void gioca() {
     for (int i = 0; i < TURNI; i++)  giocaTurno();
     calcolaRisultato(al.toString(), al.punti);
     calcolaRisultato(jack.toString(), jack.punti);
     calcolaRisultato("Totale: ", al.punti + jack.punti);
  }

 private void calcolaRisultato(String messaggio,
                                             int punti) {
     float risultato;
     if (punti == 0)
        risultato = 0;
     else
        risultato = (float) punti / (PREMIO * TURNI) *
                                                      100;
     System.out.println(messaggio + ": " + risultato);
  }
  <...>
}

private void giocaTurno() {
  Mossa mossaAl = al.gioca();
  Mossa mossaJack = jack.gioca();
  al.mossaAvversaria(mossaJack);
  jack.mossaAvversaria(mossaAl);
  if (mossaAl == Mossa.C && mossaJack == Mossa.C)
                                                        {
     al.punti += PREMIO;
     jack.punti += PREMIO;
  } else if (mossaAl == Mossa.T && mossaJack ==
                                              Mossa.T) {
     al.punti += PUNIZIONE;
     jack.punti += PUNIZIONE;
  } else if (mossaAl == Mossa.C && mossaJack ==
                                              Mossa.T) {
     al.punti += FREGATURA;
     jack.punti += TENTAZIONE;
  } else if (mossaAl == Mossa.T && mossaJack ==
                                              Mossa.C) {
     al.punti += TENTAZIONE;
     jack.punti += FREGATURA;
  }
  System.out.println(mossaAl.toString() +
                              mossaJack.toString() + " ("
        + al + ": " + al.punti + ", " + jack + ": " +
                                       jack.punti + ")");
}
```

ALCUNI DILEMMI

Non ci rimane che estendere la classe Prigioniero, simulando il c aso in cui il giocatore muova in modo random, oppure sia sempre collaborativo oppure sia vendicativo

```
public class Casuale extends Prigioniero {
  public Mossa gioca() {
     if (Math.random() >= 0.5)
        return Mossa.C;
     else
        return Mossa.T;
  }
}
public class Santo extends Prigioniero {
  public Mossa gioca() {
     return Mossa.C;
  }
}
```

il nostro main sarà il seguente

```
public static void main(String[] args) {
  DilemmaDelPrigioniero d = new
                               DilemmaDelPrigioniero(
     new Casuale(), new Casuale());
  d.gioca();
}
```

CONCLUSIONI

Abbiamo proposto un paio di strategie, ma ancora non abbiamo elaborato un vincitore, vi lasciamo il compito di trovarlo. Esistono strategie diverse e soluzioni diverse. Indagate pure su l dilemma del prigioniero e fateci sapere attraverso il nostro forum quali sono le soluzioni da voi individuate e come possono essere applicate.

Paolo Perrotta

ALGORITMI GREEDY

GLI ALGORITMI GREEDY GARANTISCONO PER SVARIATE CLASSI DI PROBLEMI UNA SOLUZIONE. SPESSO PRODUCONO SOLUZIONI POLINOMIALI PER PROBLEMI COMPLESSI, COLLOCANDOSI NELLA CATEGORIA DEGLI NP-COMPLETI.

Il modo di affrontare un problema ne individua la tecnica. Tra queste pagine abbiamo esplorato molte delle tecniche utilizzate nel mondo della programmazione. È arrivato il turno degli algoritmi greedy. Tali algoritmi ricercano la soluzione finale scegliendo ad ogni passo una soluzione ottimale. Ad ogni passo quindi, si procede scegliendo l'opzione più appetibile, più golosa (dalla traduzione del termine greedy deriva il sinonimo "algoritmo goloso"). Sgomberiamo subito il campo da pericolosi equivoci. Come vedremo con esempi, la scelta ad ogni iterazione della soluzione ottimale non garantisce la soluzione ottima finale. Ad ogni modo vedremo che si tratta di un metodo prezioso poiché, anche se non ottima, in molte situazioni viene garantito il risultato in tempi polinomiali e quindi con tempi di computazione accettabili. L'analisi della complessità e lo studio di alcuni esempi applicati al metodo verranno presentati nei prossimi paragrafi.

ALGORITMO GREEDY

Il problema di Knapsack può essere risolto attraverso un algoritmo greedy. Si tratta del problema del ladro che avendo un sacco con una capacità massima di peso oltre la quale il sacco si rompe deve rubare degli oggetti preziosi da un fissato insieme ognuno dei quali ha un valore ed un peso. Ovviamente, lo scopo è quello di massimizzare il valore della refurtiva badando a non violare il vincolo del peso, ossia non superare la capacità massima dello zaino. Anche il problema del commesso viaggiatore trova una soluzione con l'algoritmo greedy. Questo è il problema di un commesso viaggiatore che deve effettuare delle consegne in determinati luoghi e quindi deve passare per ognuno di essi, cercando di trovare il percorso minimo. Qui, il vincolo è che si debba passare una sola volta per ognuno dei luoghi. Per il problema del ladro un approccio "goloso" consiglierebbe di mettere nel sacco, purché ci entri, ad ogni passo dell'algoritmo l'oggetto di massimo valore. Facciamo un esempio. Supponiamo che il ladro abbia una sacca si capacità 20 Kg e che gli oggetti siano 4, con valore e peso come descritto dalla tabella riportata di seguito.

Valore	Peso Kg
100	15
60	8
50	9
10	2

Secondo la logica greedy si inserisce nel sacco prima il prezioso di maggiore valore: 100, dal peso 15. Successivamente, si provano a inserire gli altri oggetti di valore minore 60, e 50, ma entrambi non soddisfano il vincolo del peso, cosicché si opta per l'oggetto di minore valore 10 con peso 2. In definitiva il valore complessivo del maltolto è 110 e il peso 17. È utile far notare che un simile approccio non sempre fornisce soluzione ottima. Nel caso specifico una soluzione alternativa è inserire nello zaino gli ultimi tre oggetti che hanno peso 19, ma valore maggiore del precedente 110, esattamente 120.

Quindi in generale un algoritmo greedy può essere descritto dai seguenti punti:

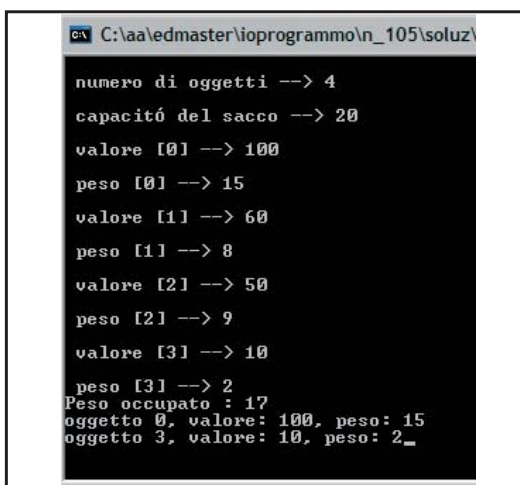
1. Trovare ad ogni passo un insieme di soluzioni ammissibili;
2. Stabilire una funzione per individuare la migliore soluzione;
3. Scegliere tra le soluzioni la migliore (la più golosa);
4. Verificare gli eventuali vincoli;
5. Definire un criterio di uscita;

La schematizzazione sicuramente rende più comprensibile le attività da svolgere. Inoltre, ci suggerisce

COSA VUOL DIRE RCPSP?

RCPSP acronimo di Resource Constrained Project Scheduling Problem, rappresenta una classe di problemi che ha come obiettivo la minimizzazione della durata del progetto in presenza di vincoli sulle risorse e di vincoli di precedenza. In generale risolvere un RCPSP significa ricercare uno schedulatore ottimo per le attività del progetto che soddisfi i vincoli di precedenza e provveda ad una corretta allocazione delle risorse disponibili quali: manodopera, materie prime, budget macchinari, equipaggiamenti, e altro ancora. La pratica dimostra che problemi reali, per progetti anche di media grandezza portano ad esaminare un elevato numero di variabili che rendono il problema immediatamente di elevata complessità, il che lo colloca tra gli NP-Hard. Si affronta quindi il problema ricorrendo ad approcci euristici che spesso costringono a sacrificare la qualità della soluzione a scapito di tempi di esecuzione accettabili.

alcune considerazioni. Al punto 2 la funzione per individuare la soluzione più “ghiotta” può definire il vero discriminante dell’efficienza dell’algoritmo. Ad esempio, per il problema del ladro appena descritto non è scontato che la funzione debba essere il maggior valore. Si potrebbe ad esempio studiare una funzione che tenga conto anche del peso.

PURO KNAPSACK GREEDY

Una soluzione greedy di Knapsack così come descritto può essere facilmente implementata. Facciamolo in C++. La struttura dati dovrà prevedere due vettori per contenere i pesi e i valori degli n oggetti e un altro vettore di booleani che indica se un oggetto si trova o meno nel sacco. Variabili globali sono: il peso attuale degli oggetti nel sacco (peso); la capacità massima del sacco (cap); il numero di pezzi rubati (pezzi); e il numero totale di oggetti (n).

```
int main()
{
// struttura dati e variabili globali
const int nmax=100;
int pesi[nmax];
int valori[nmax];
bool nel_sacco[nmax];
int peso, cap, pezzi, n;
// variabili di lavoro
int max, imax, i;
```

Dopo aver letto i vettori pesi e valori e le variabili cap e n ed aver inizializzato a false il vettore nel_sacco, l’algoritmo prevede un ciclo esterno ad ogni iterazione del quale viene collocato un oggetto all’interno del sacco. Si uscirà dal ciclo quando non sarà possibile collocare alcun elemento, oppure quando siano stati collocati ormai tutti (eventualità questa degenere) i preziosi . Si suppone come logica suggerisce che i valori siano tutti positivi, così sarà sufficiente inizializzare ad ogni iterazione il valore massimo pari a zero. Se il valore osservato è maggiore di max e non è dentro lo zaino e il suo peso non sfonda il sacco stesso allora lo si pone al suo interno. Si aggiorna il vettore di booleani il peso attuale e il numero di pezzi. Se alla fine del ciclo interno la variabile max è rimasta a zero vorrà dire che non è stato possibile collocare alcun oggetto nel sacco. Con questa condizione usciamo anche dal ciclo principale.

```
//  .... Input
pezzi=0;
peso=0;
while ( (pezzi<=n) && (max!=0))
{   max=0; // si suppongono valori positivi
   for (i=0; i<n; i++)
   if ((!nel_sacco[i]) && (valori[i]>max) && ((pesi[i]+p
                                                      so)<=cap))
   {   max=valori[i];
       imax=i;
   };
   if (max!=0)
   {  nel_sacco[imax]=true;
      peso=peso+pesi[imax];
      pezzi++;
   }
};

// ... Output
};
```

```
C:\aa\edmaster\ioprogrammo\n_105\soluz\
numero di oggetti --> 4
capacitó del sacco --> 20
valore [0] --> 100
peso [0] --> 15
valore [1] --> 60
peso [1] --> 8
valore [2] --> 50
peso [2] --> 9
valore [3] --> 10
peso [3] --> 2
Peso occupato : 17
oggetto 0, valore: 100, peso: 15
oggetto 3, valore: 10, peso: 2_
```

Fig. 1: *”Output del programma greedy knapsack”*

La soluzione ha una complessità O(n*n), ma con alcuni accorgimenti sul calcolo del massimo può ancora migliorare. Il quadrato di n si ha per la presenza del doppio ciclo innestato.

KNAPSACK FRAZIONARI

L’accezione 0/1 che si trova in letteratura associata a Knapsack problem indica la variante più diffusa, e cioè quella che prevede una logica binaria rispetto ad ogni oggetto, ossia se è presente o meno nel sacco (uno o zero). Qualora sia lecito considerare una parte di oggetto si ha a che fare con la variante frazionaria di Knapsack problem.

UNA SOLUZIONE DINAMICA

Un’altra soluzione prevede l’esplorazione di tutte le soluzioni. Si tratta di individuare tutte le possibili combinazioni. Un puro problema di calcolo combinatorio. Quindi scegliere quella che massimizzi il valore. Tale soluzione presenta una complessità elevata considerata la natura esponenziale del calcolo combinatorio da effettuare. Più precisamente si riscontra O(2n). Esistono modi alternativi per risolvere il problema knapsack, tra questi interessante è la soluzione con

programmazione dinamica. Esaminiamolo in modo da fare qualche confronto. Il nodo della questione è individuare ed esplorare in modo opportuno i sottoinsiemi di dimensione m, possibili soluzione ottime. La dimensione m del sottoinsieme ottimo non conta in fase di sviluppo dell'algoritmo. I due indici i e k rispettivamente generalizzeranno cap e n, e verranno usati nelle istruzioni cicliche per scorrere negli intervalli che tali valori definiranno. Il metodo usato è definito ricorsivamente. Una matrice raccoglie i dati parziali, al termine dell'elaborazione conterrà anche la soluzione; di nome sp. Ecco come si ottiene ricorsivamente un generico valore di sp. Esso è assegnato in funzione della condizione d[k]>i come segue.

```
sp[k, i]=sp[k-1, i]    se d[k]>i
sp[k, i]=max(sp[k-1,i], sp[k-1,i-d[k]]+ v[k] )
                                        negli altri casi
```

La riga k della matrice indica che stiamo elaborando la soluzione con k oggetti nel sacco, cosicché siamo in grado di trovare le soluzioni ottime per ogni k. L'implementazione della definizione matematica è a questo punto scontata. Tralasciando l'ovvia dichiarazione e l'input dei vettori dei pesi e dei valori, possiamo scrivere:

```
...
for (i=0; i<cap; i++)
    sp[0][i]=0;
for (k=0;  k<n; k++)
{
    sp[k,0]:=0;
    for (i=0; i<cap; i++)
      if (d[k]<=i)  // Può far parte della soluzione
                    if (v[k]+sp[k-1, i-d[k]]>sp[k-1,i])
           sp[k,i]=v[k]+sp[k-1,i-d[k]]
                    else sp[k,i]=sp[k-1,i]
             else sp[k,i]=sp[k-1,i] // Se d[k]>i
}
...
```

Il ciclo su k individua le soluzioni ottime per ogni ampiezza del sottoinsieme di soluzioni (numero di oggetti nel sacco). Il ciclo interno su i scorre fino ad arrivare alla capienza del sacco, ad ogni iterazione verifica se un generico oggetto può far parte della soluzione; in caso affermativo mediante un nuovo controllo, trova la soluzione più vantaggiosa in termini di valori. L'analisi della complessità evidenzia che la prima fase di azzeramento consta O(Cap) così come il secondo for su i. Questo ultimo ciclo si trova innestato in un altro ciclo di n iterazioni. Quindi l'intero blocco sarà O(n*Cap) che ovviamente risulta essere un risultato migliore di O(2n). Ad ogni modo non è confortante in termini computazionali il fatto che la complessità dipenda da cap (ampiezza del sacco), questo porta ad una significativa dipendenza delle prestazioni dell'algoritmo dai dati usati. Altri metodi invece non soffrono tale fattore negativo.

PROBLEMA DEL COMMESSO VIAGGIATORE

È uno dei problemi più conosciuti nel mondo dell'informatica. Dall'inglese Travelling Salesman Problem (TSP). Vi sono molti modi per formularlo. Uno dei migliori, oltre quello precedentemente introdotto, fa uso dei grafi. Dato un grafo costituito da n nodi e da un insieme di archi pesati, si vuole trovare un cammino tale che tutti i nodi siano visitati una sola volta (si parla di ciclo Hamiltoniano) e il costo sugli archi sia minimo. Questa ultima condizione equivale a dire trovare il percorso minimo se il peso dell'arco è la distanza. Prima ancora di proseguire vediamo qualche altre formulazioni. Il problema dello schedulatore, o meglio quello di sequenziare n lavori su una macchina non è altro che TSP dove i nodi sono le attività da svolgere e gli archi i tempi di attesa o set-up per risolverle. Nelle applicazioni di foratura o di rifinitura automatica eseguite da robot, i nodi sono pezzi da rifinire o fori (di varie dimensioni) da praticare, e il costo dell'arco include i necessari tempi morti. La teoria sul problema è molto ricca e richiederebbe molto spazio per essere esplorata nei suoi punti fondamentali. Ma vediamo le cose essenziali e soprattutto l'attinenza con l'oggetto dei nostri interessi, gli algoritmi greedy. Il problema può essere asimmetrico e simmetrico a seconda se il grafo è orientato o meno. Consideriamo che il grafo non sia simmetrico. La soluzione ottimale al problema si ottiene se si esaminano tutte le possibili soluzioni e tra queste si sceglie quella a costo minore, anche se impraticabile. In tal caso se consideriamo che il grafo sia completo, ovvero che ogni nodo è connesso ad ogni altro nodo ci rendiamo subito delle insormontabili difficoltà. Infatti, i possibili percorsi sono esattamente il fattoriale di n-1. Tale numero fornisce soluzioni intrattabili per n che superano numeri a due cifre, figuriamoci per reti che prevedono mappe di nazioni reali per le quali vi sono centinai di città. Bisogna quindi aggiustare la mira. Una buona soluzione è fornita applicando algoritmi greedy. Questo per un grafo completo può essere così descritto

1. Parto da un nodo
2. Valuto tutte le distanze dal nodo dove mi trovo a tutti gli altri nodi non visitati
3. Svelgo la distanza minima
4. Ripeto il punto due fin quando non sono terminati i nodi.

Altre soluzioni appartengono alla famiglia delle euristiche, ossia soluzioni che non garantiscono la soluzione ottimale ma una soluzione che si avvicina a quella ot-

timale con un certo grado di probabilità stimabile. Ma soprattutto garantiscono tempi di computazione accettabili. Si pensi ai navigatori satellitari. Ad un'interrogazione dell'utente non sono ammissibili tempi di risposta eccessivi. In tal caso si usano algoritmi euristici. Ne esistono di diversi. Euristiche costruttive come nearest neighbour che usa metodi greedy proprio perché ad ogni passi si sceglie tra i nodi più vicini. Il comportamento è mediamente buono, anche se è sempre trovare un esempio negativo per il quale il metodo si comporta in modo catastrofico. Euristiche con raffinamenti successivi, fanno parte di questa famiglia gli algoritmi Pairwise exchange, Lin e Kernighan, e k-opt. Interessanti sono anche i raffinamenti casuali che si rifanno ai processi markoviani. Va infine citato l'algoritmo di Christofides che ipotizza che sia sempre verificata la disuguaglianza triangolare che in fin dei conti non è un vincolo molto restrittivo. In tal caso in tempo polinomiale si produce un percorso al più 1.5 volte quello ottimo. È stato dimostrato che TSP è un problema NP-difficile e la versione decisionale del problema ("dati i pesi e un numero x, decidere se ci sia una soluzione migliore di x") è NP-completa. Il problema rimane NP-difficile anche in molte sue versioni ridotte, come nel caso in cui le città siano in un piano con distanze euclidee. Inoltre, omettere la condizione di visitare una città "una e una sola volta" non rimuove la NP-difficoltà, poiché si nota facilmente che nel caso piano un cammino ottimale visiterebbe comunque le città una volta sola.

PROBLEMI P E NP

Sappiamo che la teoria della complessità computazionale è quella parte della teoria della computazione che studia la quantità di risorse richieste durante una computazione per risolvere un dato problema. Le risorse oggetto di studio sono lo spazio (quanta memoria è necessaria per risolvere un problema) e il tempo (quanti passi- istruzioni elementari- sono necessarie a risolvere un problema). Se il tempo impiegato da una macchina deterministica è polinomiale rispetto alla dimensione della struttura dati di input si ha la classe P; la classe NP, volendo semplificare, consiste di tutti quei problemi di decisione le cui soluzioni positive possono essere verificate in tempo polinomiale avendo le giuste informazioni, o equivalentemente, la cui soluzione può essere trovata in tempo polinomiale con una macchina non deterministica (la macchina di Turing non deterministica si distingue da quella deterministica definita in precedenza per il fatto che, in presenza di un determinato stato e di un determinato carattere letto, essa permette più transizioni). Un problema da un milione di dollari (sul serio) è sapere se: P è uguale a NP? Quando nel 1971 fu introdotto il problema si era molto ottimisti sulla possibile soluzione in tempi brevi. Oggi la maggior parte degli studiosi è orientata dare risposta no, mentre più pragmaticamente altri pensano che la domanda possa essere indecidibile rispetto agli assiomi correntemente accettati. Il grande logico Kurt Gödel, invece, in una lettera a John von Neumann, riteneva plausibile l'ipotesi P=NP. Un premio di un milione di dollari è stato offerto per la soluzione corretta (si tratta di uno dei problemi del millennio).

Un ruolo importante in questa discussione è giocato dall'insieme dei problemi NP-completi, che semplificando possono essere descritti come quei problemi in NP che meno probabilmente appartengono anche a P. Gli informatici teorici hanno dimostrato che se P è diverso da NP, allora nessun problema NP-completo appartiene a P. La letteratura si è arricchita molto sulla questione, sono state introdotte infatti importanti classificazioni. È stato osservato che molti dei problemi che stavano nella classe NP hanno una stessa struttura: la costruzione della soluzione con un algoritmo non deterministico e la verifica della soluzione costruita con un algoritmo deterministico. Ci si chiedeva quindi se ci fossero elementi comuni in questi problemi, e in effetti c'erano. Sono stati individuati problemi molto particolari. Un algoritmo per risolvere uno di questi problemi può essere convertito in un algoritmo per risolvere un qualunque problema NP. Questi problemi sono stati detti NP-difficili (NP-hard). Un problema NP-difficile potrebbe anche non essere appartenente all'insieme NP, nel senso che la verifica della soluzione o equivalentemente l'"algoritmo di Gastone" (ogni volta che si deve fare una scelta, si indovina sempre la strada corretta) potrebbe richiedere un tempo più che polinomiale. Per dimostrare questa sorta di equivalenza, ci si riconduce alla teoria dei linguaggi, e si sfrutta il concetto di riduzione.

È interessante notare che quasi tutti i problemi NP sono anche NP-completi; l'unica eccezione nota, per ora, è l'isomorfismo di grafi, per il quale nessuno è ancora riuscito a dimostrare né la completezza, né l'eventuale appartenenza alla classe P. Ricordo anche la primalità dei numeri trattata tra queste pagine ha avuto una brillante dimostrazione nel 2002, che spostava il problema in P. Il problema del commesso viaggiatore come detto è NP-completo. Un tentativo interessante è la ricerca di un algoritmo polinomiale per la soluzione di uno qualunque dei problemi NP-completi: questo farebbe automaticamente fatto collassare tutta la classe di problemi NP nella classe P. Ovviamente nessuno è riuscito a trovarne uno, né nessuno è mai riuscito a dimostrare che P include NP, sebbene molti esperti sospettino che questa sia la relazione tra le due classi.

Fabio Grimaldi

LA SFIDA DEL MILLENNIO

In modo del tutto analogo alla epica sfida lanciata da Hilber ad inizio secolo scorso, l'istituto matematico Clay ha proposto una nuova sfida del millennio. Chiunque riuscisse a risolvere uno dei sotto elencati problemi o congetture vincerebbe il premio di un milione di dollari. Tra questi spicca il problema tutto informatico P vs NP.

- **P contro NP**
- **Congettura di Hodge**
- **Congettura di Poincaré**
- **Ipotesi di Riemann**
- **Teoria di Yang-Mills**
- **Equazioni di Navier-Stokes**
- **Congettura di Birch e Swinnerton-Dyer**

Chi ne vuole saper di più può riferirsi al seguente indirizzo http://www.claymath.org/ e proseguire alla voce awards\millenium